ISO/IEC JTC 1/SC 18 WG8 N1920rev

TITLE: **Information processing -- Hypermedia/Time-based Structuring Language (HyTime) - 2d edition**

SOURCE: **WG8**

PROJECT: **JTC1.18.15.1**

PROJECT EDITORS: **Charles F. Goldfarb, Steven R. Newcomb, W. Eliot Kimber, Peter J. Newcomb**

STATUS: **Approved text**

ACTION: **For information**

DATE: **9 May 1997**

DISTRIBUTION: **WG8 and Liaisons**

REPLY TO: **Dr. James David Mason**
(ISO/IEC JTC1/SC18/WG8 Convenor)
Lockheed Martin Energy Systems
Information Management Services
1060 Commerce Park, M.S. 6480
Oak Ridge, TN 37831-6480 U.S.A.
Telephone: +1 423 574-6973
Facsimile: +1 423 574-0004
Network: masonjd@ornl.gov
http://www.ornl.gov/sgml/wg8/wg8home.htm
ftp://ftp.ornl.gov/pub/sgml/wg8/

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 10744 was prepared by Joint Technical Committee JTC1, *Information technology*.

Annexes A, B, and C form an integral part of this International Standard. Annex D is for information only.

Introduction

The Hypermedia/Time-based Structuring Language (HyTime), defined in this International Standard, provides facilities for representing static and dynamic information that is processed and interchanged by hypertext and multimedia applications. HyTime is an application of ISO 8879, the Standard Generalized Markup Language (SGML).

HyTime supports the classic bibliographic model of information referencing, whereby it is possible to represent links to anything, anywhere, at any time, in a variety of ways. The extension of this model to the computerized information age, known as “integrated open hypermedia” (IOH), is the field of application of HyTime.

HyTime provides standardized mechanisms for specifying interconnections (hyperlinks) within and between documents and other information objects, and for scheduling multimedia information in time and space.

Without HyTime, such information is typically embedded in the processing instructions of hypermedia “scripts” that govern the rendition of such documents, and is therefore not usable for other forms of processing. When HyTime is used, those properties of the information that are independent of specific processing are available for processing by applications and platforms other than the one on which the information was created.

It is for the application designer and user to decide which properties can be isolated from the scripts in this way. In an ideal world, the sole consideration would be whether the properties are intrinsic to the information, regardless of how it is processed. For example, the title of this clause is intrinsic information; the font that it appears in normally is not.

In the real world, representation strategies will vary from one situation to another and will depend on such other considerations as the expected uses of the information, the flexibility of the scripting language, and performance considerations. For this reason, HyTime is highly modularized so that application designers need use only the facilities for the properties they care to describe in a standardized way.

HyTime's rules for the standardized expression of hypermedia structuring are expressed as an “enabling architecture”, consisting of a number of “architectural forms” and their associated semantics. The HyTime standard's formal definition as an architecture conforms to the Architectural Form Definition Requirements in annex A of this International Standard.

0.1 HyTime modules

The architectural forms and attributes of the HyTime language are grouped into five modules, each of which have both required and optional facilities. Support for the modules and their options is indicated by “HyTime support declarations.”

— Base module

The base module consists of independent utility facilities, some of which are optional. The required facilities support hyperdocument management (using SGML) and identification of object properties. The optional facilities provide lookup tables for commonly used elements, a mechanism for associating use and access policies with objects, and a mechanism for relating attributes and the content of elements to their semantic values by reference. The base module also defines the fundamental coordinate addressing notation used by all the other HyTime modules.

— Location address module

The location address module allows the identification of objects that cannot be addressed by SGML unique identifiers, and objects that are in external documents.

Three basic types of address may be supported: name, semantic location, and coordinate location. Addressing of multiple locations is also possible. The syntax and semantics of these addressing mechanisms are independent of the data content notations of the data being addressed.

NOTE 1 The ability to resolve HyTime addresses in a given notation is dependent on software that can interpret that notation in terms of the abstractions HyTime uses for all addressing (see *6.1.1 Object representation*).

HyTime's system- and notation-independent way of expressing addresses of hypermedia objects also provides the basis of its hyperlinking and scheduling power.

— Hyperlinks module

This module allows relationships (“hyperlinks”) to be established among objects, either within a single document or among the constituent documents and information objects of a hyperdocument.

— Scheduling module

This module allows events — occurrences of objects — to be scheduled on the coordinate axes of “finite coordinate spaces” in such a way that their positions can be expressed in terms of their relationships to one another. Measurement along the coordinate axes can be in terms of spatial or temporal units.

— Rendition module

When the scheduling module is used, object modification and/or event projection can be used to represent parameters governing the rendition process.

- Object modification

  The object modification facility allows specification of the order in which objects are to be modified during rendition and of the “object modifiers” (such as amplifiers and filters) that will affect them.

  NOTE 2 The semantics of the modifiers are not defined by HyTime.

- Event projection

  Rendition requires the projection of events into a coordinate space where they can be perceived; for example, from a coordinate space with a virtual time axis to one based on real time. The event projection facility allows specification of the factors for computing the positions and sizes of events in the target coordinate space.

  In situations where the rendered position and size of an event is not predictable (as when user interaction will affect it), the virtual dimensions of the original events may be projected onto real space/ time via a formula in some arbitrary user-defined expression language. Such an expression may, among other things, accept late-binding values during rendition to resolve the positions and sizes of projected events.

  NOTE 3 The semantics of formatting the objects to fit the new extents is not defined by HyTime.

Applications can choose to include rendition information as an essential part of a hyperdocument, or it can be incorporated in the “style sheets” of the processing programs. The choice depends on the nature of the information being rendered. In multimedia documents, for example, rendition style tends to be more essential to the document than is the case in conventional documents.

0.2 HyTime applications

HyTime provides a generic level of support for a variety of applications, rather than the semantics for a specific application (that is, HyTime is like a carrier or infrastructure).

The boundary between an application and HyTime is variable, and is determined by the application designer, who is free to decide how much of the information will be expressed in a standardized way using HyTime and how much will be application-specific (for example, in a data content notation).

Because the semantics of HyTime's architectural forms and attributes are standardized, it will be possible to implement supporting software and/or hardware usable for a variety of applications. Applications can define additional attributes when defining an element type that is based on an architectural form. The semantics of the application-defined element types and attributes are the only ones defined by the applications themselves. They could be standardized by an industry group or formally by a national or international standards body.

HyTime attributes have no intrinsic meaning other than that specified in this International Standard. However, an application can impute additional semantics to them, either implicitly, or by defining appropriate element types and attributes. For example, to HyTime, the “dimension reference” architectural form means only that the dimension of one object is calculated from the dimension of another. An application, however, could specify (if it wished) that use of dimension referencing implies a synchronization relationship between the objects, and could emphasize this by using “sync” as the generic identifier of a dimension reference element type.

HyTime elements can occur wherever an application's DTD and the HyTime meta-DTD allow. A finite coordinate space could occur, for example, within a paragraph of a memo in order to represent a calendar or project plan in that context, or several paragraphs could occur as the content of a timed event.

Clients of HyTime, including applications and application architectures, can define non-HyTime architectural forms as well as elements. Although an application may not add architectural forms to HyTime, nor combine HyTime architectural forms with one another, it can create its own architecture (for example, “MyArch”) defining its own set of architectural forms. These architectures may be derived wholly or in part from the HyTime architecture. The facilities for defining and using architectures are defined in annex A.3.

If, for example, a document is derived from the HyTime and MyArch architectures, after the content and attributes of each element are processed and validated in SGML terms by the SGML parser, elements with HyTime attributes would be subject to processing and validation by the HyTime engine, while elements with MyArch attributes would be subject to appropriate processing and validation by the application, perhaps aided by a MyArch engine.

HyTime defines some of the parameters needed by an application to accomplish rendition, and some of the rendition functionality. The remainder is provided by the application, or by a document architecture to which the application conforms.

Many different HyTime-conforming applications and architectures could exist, to address different requirements and serve different user constituencies. Such architectures could be incompatible in their non-HyTime aspects, but would still be supportable by a single HyTime engine.

NOTE 4 For example, no application would need to invent its own system for representing finite coordinate spaces, even if its projection functions were extremely intricate and application-specific. HyTime allows application-specific projection functions, using application-chosen (or defined) function languages, to be represented in conjunction with standardized representations of the unprojected and projected finite coordinate spaces.

HyTime's design is optimized for the sequencing and alignment problems encountered in typical hypermedia applications; it is not intended as a general architectural solution for compound document page layout, for which other solutions are better suited.

NOTE 5 However, HyTime is compatible with a wide variety of such solutions. For example, HyTime finite coordinate spaces could be used to describe the media onto which page description language objects are imaged.

NOTE 6 HyTime shares with the DSSSL standard (ISO/IEC 10179:1996, Document Style Semantics and Specification Language) the fundamental SGML property set and grove abstraction for representing and operating on parsed SGML documents (and other data objects for which groves can be constructed).

0.3 Organization of this document

The structure of this document reflects the modular structure of HyTime, as follows:

- The base module clause (clause 6) is a prerequisite for the other clauses. Some of the facilities it describes are required for all uses of HyTime.
- The location address (clause 7), hyperlinks (clause 8), and scheduling (clause 9) clauses describe modules that are independent of one another.
- The rendition clause (clause 10) describes a module that is dependent on the scheduling module.
- The conformance clause (clause 11) describes requirements that apply to all conforming HyTime documents, applications, and systems.

The text of this International Standard also includes the following annexes:

- annex A

  This normative annex defines the SGML Extended Facilities, many of which are prerequisites for the other clauses.
- annex B

  This normative annex defines the HyTime property set.
- annex C

  This normative annex contains the complete HyTime and General Architecture meta-DTDs as they are used by architectural engines.
- annex D

  This informative annex identifies sources of supplementary tutorial and reference materials for HyTime.

1 Scope

1.1 Definition of scope

This International Standard defines a language and underlying model for the representation of “hyperdocuments” that link and synchronize static and dynamic (time-based) information contained in multiple conventional and multimedia documents and information objects. The language is known as the “Hypermedia/Time-based Structuring Language”, or “HyTime”.

HyTime can represent time in both the abstract, or “musical” sense, and in user-defined real-time units. It also provides a way of relating the two so that elements of time-dependent documents can be synchronized.

NOTE 7 This facility extends to the representation of multimedia information the power, once limited to conventional documents, to distinguish intrinsic information content from style considerations.

HyTime's techniques for representing its time model are equally applicable to spatial and other domains; all are treated as systems for measuring along different axes of a coordinate space. Arbitrary cross-references and access paths based on external interactions (“hypermedia links”) are also supported.

HyTime's time representation contains sufficient information to derive the durations of both control (“gestural”) data (e.g., control information for audio or video hardware) and visual data (e.g., a music score, presentation storyboard, or television script).

The media formats and data notations of objects in a HyTime hyperdocument can include formatted and unformatted documents, audio and video segments, still images, and object-oriented graphics, among others. Users can specify the positions and sizes of occurrences of objects in space and time, using a variety of measurement units and granularities. Temporal requirements of applications ranging from animation to project management can be supported by choosing appropriate measurement granules.

NOTE 8 This International Standard does not address the representation of audio or video content data, but simply defines the means by which the start-time and duration of such data can be synchronized with other digitized information. Nor does it specify the layout process by which occurrences of unformatted documents and other information objects can be made to fit the positions and sizes specified for them.

HyTime is an enabling standard, not an encompassing one. As a result, the objects comprising a HyTime hyperdocument are free to conform to any application architectures, or to document architectures imposed by standards, and to be represented in any notation permitted by those architectures. Only the “hub document”, which may determine the hyperdocument membership, must conform to HyTime in addition to any other architectures to which it may conform.

HyTime is designed for flexibility and extensibility. Optional subsets can be implemented, alone or in conjunction with user-defined extensions.

The Hypermedia/Time-based Structuring Language (HyTime) is an SGML application conforming to International Standard ISO 8879 — Standard Generalized Markup Language.

The hyperdocument interchange format recommended in this International Standard is ISO 9069, the SGML Document Interchange Format (SDIF). SDIF is defined in Abstract Syntax Notation 1 (ISO 8824) and can be encoded according to the basic encoding rules of ISO 8825 for interchange using protocols conforming to the Open Systems Interconnection (OSI) model. Other interchange formats can also be used.

1.2 Field of application

The field of application of HyTime is “integrated open hypermedia” (IOH), the “bibliographic model” of hyperlinking wherein an author can, by a suitable reference, link to anything, anywhere, at any time.

Because of HyTime's modular design and flexible conformance rules, implementations need support only those facilities that are within their present capabilities. User investment in hyperdocument preparation is nevertheless encouraged because of the well-defined upward-compatible path to a full hypermedia solution.

HyTime is intended for use as the infrastructure of platform-independent information interchange for hypermedia and synchronized and non-synchronized multimedia applications. Application developers will use HyTime constructs to design their information structures and objects, and the HyTime language to represent them for interchange.

NOTE 9 The HyTime language is not intended for encoding the internal representation of information on which application programs act while executing.

Applications can use HyTime to represent hyperdocuments containing information that is at any stage of rendition, from “revisable” to “optimized for interactive access”. An application can also choose to convert a rendition of a HyTime hyperdocument into an optimized form for transmission or interactive presentation.

NOTE 10 Whether the HyTime representation of a hyperdocument can be used in a local file system for direct access by programs will depend on the type of information in the hyperdocument, the speed of the platform, and the functions performed by the applications that access the hyperdocument.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 31-0:1992, *Quantities and units — Part 0: General principles*.

ISO 3166:1993, *Codes for the representation of names of countries*.

ISO 8879:1986, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*.

ISO 9069:1988, *Information processing — SGML support facilities — SGML Document Interchange Format (SDIF)*.

ISO/IEC 9070:1991, *Information processing — SGML support facilities — Registration procedures for public text owner identifiers*.

ISO/IEC 10179:1996 *Information technology — Processing languages — Document Style Semantics and Specification Language (DSSSL)*.

ISO/IEC 13673:1997, *Information technology — Text and Office Systems — Conformance Testing for Standard Generalized Markup Language (SGML) Systems*.

3 Definitions

For the purposes of this International Standard, the following definitions apply.

3.1
anchor

An object (or list of objects) that is linked to other objects or lists of objects by a hyperlink.

NOTE 11 The term “object” is not a formal construct in HyTime; an anchor could be a document, an element, an arbitrary chunk of data, or any other thing.

NOTE 12 An object is an anchor if and only if a hyperlink identifies it as such.

NOTE 13 An object could be identified as an anchor in several link ends, in the same or different hyperlinks.

3.2
anchloc

Hyperlink anchor location address.

3.3
application BOS

A bounded object set that is determined by an application according to its own rules.

NOTE 14 The rules can provide for automatic determination from a set of parameters, selection by the user, or some combination thereof. One possibility is for the application to use the entities included in a HyTime BOS as a starting point, but to allow the user to add or subtract entities from it.

3.4
architectural forms

Rules for creating and processing components of documents. There are four kinds: element form, attribute form, notation form, and data attribute form.

3.5
attribute form

An architectural form that applies to attributes of elements.

3.6
auxiliary grove

A grove constructed by processing nodes in another grove.

NOTE 15 For example, the data tokenizer grove constructed for use by a data location address is an auxiliary grove.

3.7
bit combination

An ordered collection of bits (for example, a byte is a combination of 7 or 8 bits). A bit combination represents a character in character data or markup, but can represent numeric or other values in non-character data.

3.8
bounded object set
BOS

The subject that a HyTime application processes: a set of one or more documents and other information objects.

NOTE 16 There are three kinds: HyTime BOS, application BOS, and effective BOS.

3.9
bounding region

The region of an FCS addressed by an fcsloc within which events are selected according to the selection precision specified for the fcsloc.

3.10
children property
In groves, that subnode property of a node that is designated as the content property of the node. A node may have zero or one children properties.

NOTE 17 If a content property is not nodal, then the node does not have a children property.

3.11
client architecture
An architecture derived from another architecture. The first architecture is said to be a client of the architecture from which it is derived.

3.12
client document
A document conforming to an architecture. The document is said to be a client of the architecture to which it conforms.

3.13
client DTD
The document type definition of a client document.

3.14
clink
Contextual link element form

3.15
content property
In groves, that property of a node that is designated as containing the semantic content of the node. A node may have zero or one content properties. A content property may be nodal or primitive. When a content property is nodal, the property is also the children property of the node.

NOTE 18 For example, for SGML element nodes, the content property can contain elements, data characters, or other objects that occur syntactically within SGML elements, excluding objects that the rules of SGML exclude from the semantic content, such as ignored record ends and separators in element content.

3.16
content tree
In groves, the tree formed by a node and the nodes in its children property.

3.17
contextual hyperlink
A hyperlink that occurs “in context”, meaning that one anchor of the link is the link element itself (a “self anchor”) and is a traversal initiation anchor. In an interactive application, the self anchor can be accessed externally from adjacent elements or data in the document hierarchy.

NOTE 19 Any of the hyperlink element forms may be used contextually. The clink element form is always contextual.

3.18
contextual link element form
An element form that represents a binary contextual hyperlink having the fixed anchor roles reference mark (refmark) for the self anchor and reference subject (refsub) for the other anchor.

3.19
data attribute form
An architectural form that applies to data attributes.

3.20
data location address
A location address that addresses the string and token data objects resulting from tokenization of character data.

3.21
dataloc
Data location address.

3.22
dimension
Size and position on a coordinate axis. It consists of three components: a position (the first occupied quantum), a quantum count (the total number of quanta occupied), and the last occupied quantum.

3.23
document
A collection of information that is identified as a unit and that is intended for human perception.

3.24
document (type) definition
DTD
Rules, determined by an application, that apply SGML to the markup of documents of a particular type. A document type definition includes a formal specification, expressed in a document type declaration, of the element types, element relationships and attributes, and references that can be represented by markup. It thereby defines the vocabulary of the markup for which SGML defines the syntax.

NOTE 20 A document type definition can also include comments that describe the semantics of elements and attributes, and any application conventions.

3.25
effective BOS
The bounded object set consisting of all the objects that at any given point have been successfully and fully integrated into the hyperdocument being processed.

3.26
element form
An architectural form that applies to elements.

3.27
entity descriptor
A component of an SDIF data stream that represents an external entity.

3.28
entity tree
A tree structure whose nodes are entities, constructed by the following steps:

1) An SGML document entity or SGML subdocument entity is selected as the root node.
2) The set of external entities identified by external identifier parameters of markup declarations in the node comprise the children of the node. Each child that is an SGML document entity or SGML subdocument entity is selected as a parent node.
3) Step 2 is repeated for each parent node, until the leaves of the tree or a specified maximum number of levels is reached.

NOTE 21 The maximum number of levels is an attribute of a HyTime document when used as a hub, and can be overridden when a HyTime application is invoked.

3.29
event
The occurrence of an object in a coordinate space. It associates the object with a scheduled extent.

NOTE 22 The scheduled extent of an event gives the object a position and size. The set of first quanta on all the axes defines the position, while the set of quantum counts for all axes completes the dimension specifications and defines the size.

3.30
event projection
The conversion of the scheduled extent of an event from schedule to schedule (the “unprojected” to the “projected”).

NOTE 23 For example, from music time to real time, or from user device coordinates to real space units.

3.31
(scheduled) extent
Size and position in a coordinate space. It consists of a dimension on each axis of the space.

3.32
external identifier
A parameter of an SGML markup declaration (typically an entity declaration) that identifies an external information object.

NOTE 24 It may do so by various means, including:

— a formal public identifier; that is, a globally unique public identifier, which allows a system to access its object by means of a table look-up; and/or

— a system identifier; that is, a file identifier, storage location, program invocation, data stream position, or other system-specific means of locating the object in storage.

3.33
fcsloc
finite coordinate space location address

3.34
finite coordinate space location address
A location address that addresses events in an event schedule (or the objects scheduled by those events) by defining a bounding region enclosing the events selected.

3.35
graph representation of property values.
An abstract data structure consisting of a directed graph of nodes in which each node may be connected to other nodes by labeled arcs.

3.36
grove
Graph Representation Of property ValuEs.

3.37
grove construction process
A process that constructs a primary or auxiliary grove.

3.38
grove definition
The combination of a grove plan, grove construction process, and grove source object.

3.39
grove plan
A specification of what modules, classes, and properties to include in a grove. Grove plans are used both to construct groves and to view existing groves.

3.40
grove root
The one node in a grove that does not have an origin.

3.41
grove source
The data (for primary groves) or nodes (for auxiliary groves) from which a grove is constructed.

3.42
hub document
The document in which access to a hyperdocument begins. In a HyTime hyperdocument, the hub document also defines a HyTime BOS for interchange, rendition, or other processing.

NOTE 25 "Hub document" is not a permanent state of a document and cannot be specified by means of an attribute. The designation of a hub document is a parameter of processing and is specified when an application is invoked.

NOTE 26 It is possible that several HyTime documents, when designated as a hub document, could define the same HyTime BOS.

3.43
hyperdocument
Two or more documents or other information objects that are connected to one another by a web.

NOTE 27 Access to a hyperdocument begins from a designated hub document.

NOTE 28 One hyperdocument may include another, otherwise independent hyperdocument, by declaring the hyperdocument's hub document to be included as a "subhub".

NOTE 29 A hyperdocument may represent a fixed or singular collection of data objects organized for a particular purpose (such as to represent common ownership or to support a particular rhetorical purpose) or it may be an arbitrary and transitory collection of objects. The HyTime bounded object set control facilities and activity policy association facilities, as well as application-specific mechanisms, can be used to define additional semantics for and constraints on the composition and use of hyperdocuments.

3.44
hyperlink
An information structure that represents a relationship among two or more objects.

NOTE 30 Objects that are related by a hyperlink are called the "anchors" of the hyperlink. An anchor is identified by a property of the hyperlink called a "link end".

NOTE 31 Hyperlinks can be assigned link types and names by applications and architectures that use HyTime.

NOTE 32 An SGML document can represent relationships by means other than hyperlinks; for example, the subordinate and sibling relationships of the document hierarchy are represented by the position of markup tags.

3.45
hyperlink anchor location address
A form of query location address that addresses objects by the anchor role names of the anchors of which the objects are members.

3.46
hyperlink location address
A form of query location address that addresses hyperlinks by link type.

3.47

hypermedia application

An information processing application that has hypertext and/or multimedia capabilities.

NOTE 33 An implication of this definition is that “hypermedia” refers to the union of hypertext and multimedia, rather than their intersection.

NOTE 34 While it would be possible to maintain a strict distinction among the terms “hypertext”, “multimedia”, and “hypermedia”, there is generally little point in doing so in this International Standard when referring to documents or applications. Events, however, are referred to as “multimedia” rather than “hypermedia” because their nature is unaffected by whether hyperlinks are made to them. Similarly, hyperlinks may be characterized indiscriminately as “hypertext”, “hypermedia”, or simply “hyper”, because their nature is unaffected by the objects to which they are linked.

3.48

hypermedia document

A document or hyperdocument that is used in a hypermedia application.

NOTE 35 A hyperdocument is almost always a hypermedia document as well, in that it is normally used in a hypermedia application. However, not all hypermedia documents are hyperdocuments; for example, a document containing video clips but no hyperlinks.

3.49

Hypermedia/Time-based Structuring Language

HyTime

A standardized hypermedia structuring language for representing hypertext linking, temporal and spatial event scheduling, and synchronization. HyTime provides basic identification and addressing mechanisms and is independent of object data content notations, hyperlink types, processing and presentation functions, and other application semantics. Hyperlinks can be established to documents that conform to HyTime and to those that do not, regardless of whether those documents can be modified. The full HyTime function supports “integrated open hypermedia” (IOH) — the “bibliographic model” of referencing that allows hyperlinks to anything, anywhere, at any time, in a variety of ways — but systems need support only the subset that is within their present capabilities.

3.50

HyTime attribute

An attribute whose definition is included in a HyTime architectural form.

3.51

HyTime BOS

A HyTime BOS is a tree of entities rooted at and specified (directly and/or indirectly) by the entity declarations and other specifications in a HyTime hub document.

NOTE 36 A HyTime BOS can be determined automatically by a HyTime engine. The root of the entity tree is the SGML document entity of the hub document.

3.52

HyTime document

An SGML document whose properties are represented essentially as defined in this International Standard.

NOTE 37 A HyTime document is normally a conforming HyTime document (see *11.1 Conforming HyTime document*).

3.53

HyTime element

An instance of a HyTime element type.

3.54
HyTime element type

An element type in a HyTime document that conforms to a HyTime architectural form.

NOTE 38 The element type itself is not defined by HyTime. In addition to attributes defined by the architectural form, it may have application-specific attributes.

3.55
HyTime engine

A program (or portion of a program or a combination of programs) that recognizes HyTime constructs in documents and performs application-independent processing of them.

NOTE 39 For example, a HyTime engine can interface with data base and network servers to resolve and access anchors. It can also perform the calculations to locate events consistently in a coordinate space regardless of the schedules in which they were entered or the measurement units used to define their extents.

3.56
HyTime hyperdocument

A hyperdocument whose hub document is a HyTime document.

NOTE 40 The hub document must be represented in SGML; the other components of the hyperdocument need not be. The hub or any other component may conform to a document architecture.

3.57
HyTime system

An SGML system that includes a HyTime engine.

3.58
hypertext

Information that can be accessed in more than one order.

NOTE 41 A hypertext can be a single document or a library of documents (a “hyperdocument”). For example:

- A novel is typically not designed to be a hypertext.
- A book with footnotes or internal cross-references is a single document hypertext.
- A book with external cross-references (e.g., bibliographic citations) is a member of a library that as a whole constitutes a hyperdocument.
- A book with both internal and external cross-references is both a single document hypertext and a member of a library that as a whole constitutes a hyperdocument.

3.59
initial referrer (to a location path)

A referrer to the first location step in a location path.

NOTE 42 An initial referrer cannot be a location address because one location address that references another becomes a location step.

NOTE 43 Every step in a given location path has the same initial referrer.

3.60
integrated open hypermedia
IOH

The formalization, for computer processing, of the “bibliographic model” of referencing that allows representation of hyperlinks to anything, anywhere, at any time, in a variety of ways.

3.61
link
Depending on context, either a hyperlink or an SGML processing link (“link process”).

3.62
link end
In a hyperlink, the union of an anchor role name, the members of the anchor, and the other properties associated with the anchor, such as traversal rules.

3.63
link process definition
LPD
Application-specific rules that apply SGML to describe a link process. A link process definition includes a formal specification, expressed in a *link type declaration*, of the link between elements of the source and result, including the definitions of source attributes applicable to the link process (“link attributes”).

NOTE 44 A link process definition can also include comments that describe the semantics of the process, including the meaning of the link attributes and their effect on the process.

3.64
link type
A class of hyperlinks. It assigns meaning to the relationship represented by the hyperlinks, including the role in the relationship played by each anchor.

3.65
linkloc
Hyperlink location address.

3.66
listloc
List location address.

3.67
list location address
A location address that selects nodes from a node list by specifying one or more dimension specifications, one for each contiguous sequence of nodes selected.

3.68
location ladder
A set of location addresses, known as “location rungs”, in which each rung is the location source of the rung below it.

NOTE 45 A location ladder is visualized as running from top to bottom, from location source to path step. A rung that is a step in a location path is considered to be the bottom rung with respect to that path.

NOTE 46 A location ladder represents a progressive culling of the set of addressable objects as one proceeds downward (in other words, the top rung addresses the largest possible scope from which nodes may be selected by the rungs below it).

3.69
location address
An element form that represents the address of one or more objects. A reference to a location address is treated as a reference to the objects located by the location path that it begins.

NOTE 47 A location address can be the location source of another location address, thereby forming a location ladder.

NOTE 48 A location address is the HyTime representation of what is commonly known in computing as an “indirect address”.

3.70
location path
A set of location addresses, known as “location steps”, in which the first step locates the second and so on. A location path can have branches, created when a step locates two or more objects, at least one of which is a location address. A branch terminates when its last step locates only objects that are not location addresses. The objects located by a location path are all those, other than steps, that are located by any of the steps in the path.

NOTE 49 A location path is visualized as running from left to right, from initial referrer to objects addressed. Each step is the bottom rung of a location ladder.

NOTE 50 A location address can be a step in more than one location path, but can be the first step in only one.

3.71
location source
The set of objects from which a location address selects the objects that it addresses.

NOTE 51 For example, the location source of a treeloc is the tree in which the node addressed by the treeloc is found.

3.72
mixedloc
mixed location address
A location address that addresses objects indirectly through its child location address elements.

3.73
multimedia
(Adjective) Employing more than one means of communicating something, such as forms used by artists, musical composers, etc.

3.74
(object) modification
The modification of an object by another object during rendition.

NOTE 52 For example, routing audio signals through an effects box.

NOTE 53 HyTime deals with the scheduling and association of modifiers, but not the semantics of modification.

3.75
nameloc
Named location address.

3.76
named location address
A specialized form of mixed location address that addresses objects by their element IDs or entity names.

3.77
named node list
In groves, a node list in which all the member nodes exhibit a unique value for a common “name” property. The value of a name property may be either a string or a single node where the node is unique among all the nodes in the name values of the members of the node list.

3.78
name-space location address
A location address that addresses nodes in named node lists.

3.79
nmsploc
Name-space location address.

3.80
node
In groves, an ordered set of properties representing a single object.

3.81
notation form
An architectural form that applies to data entities and notation data content.

3.82
origin
In groves, for a node, the node of which the node is a subnode. Every node in a grove, except the grove root, has exactly one origin node.

3.83
patch
An interconnection of modifiers.

3.84
pathloc
Path location address.

3.85
path location address
A location address that addresses nodes in a tree by viewing the tree as a matrix where each column is the list of nodes from the root to a leaf.

3.86
pelement
Pseudo-element.

3.87
presentation
A state of processing of a document in which it is ready for human perception.

3.88
previous specified element
The element in the SGML representation of the document that was most recently parsed.

NOTE 54 It is typically also the one most recently occurring when the SGML representation is viewed as a character string, although an entity reference could cause other elements to be parsed more recently than that one.

3.89
primary grove
A grove constructed by processing source data.

NOTE 55 For example, the grove constructed for an SGML document after the initial parse of the SGML source data is a primary grove.

3.90
principal tree
In groves, that content tree designated as being the principal tree by the grove's property set. In HyTime, the principal tree is the default implicit location source for tree location addresses.

3.91
principal tree root
In groves, the node that is the root of the grove's principal tree. In the SGML property set, the document element is the principal tree root.

3.92
projection
Event projection.

3.93
property location address
A location address that addresses the value of a property of a node in a grove.

3.94
property set
A formal definition document, conforming to the requirements of the Property Set Definition Requirements in this International Standard, that specifies the node classes and properties to be used in constructing a grove.

3.95
proploc
property location address.

3.96
pseudo-element
In the SGML property set, an object whose children are derived from the data and other constructs occurring between a tag close and the next tag open. Pseudo-elements are children of elements.

3.97
quantum
Countable division of a coordinate axis.

3.98
queryloc
Query location address.

3.99
query location address
A location address that uses a query to address nodes in a grove by their properties.

3.100
real time
Time in the every-day sense, as measured in seconds, minutes, hours, etc.

3.101
referent (of an ID reference)
An element whose ID is a value of an ID reference or ID reference list attribute (or content).

3.102
referential attribute
An attribute of an element that is designated as being a reference either by declaring it as an IDREF, IDREFS, ENTITY, or ENTITIES attribute or through the use of the HyTime refloc facility.

NOTE 56 In the HyTime meta-DTD, referential attributes are identified by the conventional comment “-- Reference --”.

NOTE 57 Content may also be designated as referential.

3.103
referrer (to a referent element)
An element with a referential attribute (or content) that addresses the referent element.

3.104

relative location address

A location address that addresses nodes in a tree by specifying the genealogical relationship of the nodes to a starting node and then selecting nodes from the list of relatives.

3.105

relloc

relative location address.

3.106

rendition

A process performed on a HyTime document to prepare it for presentation. It may include event projection and object modification as defined by HyTime, in addition to application-specific processing.

NOTE 58 Several renditions might be performed in sequence in order to create the presentation that is ultimately perceived by a user. The renditions could involve projecting events to a new coordinate space; for example, from virtual time to real time.

3.107

reportable HyTime error

RHE

A failure of a HyTime document to conform to the requirements of this International Standard other than one that:

a) cannot be detected without processing the document; or

b) is identified in this International Standard as not being reportable.

3.108

(location) rung

An element in a location ladder.

3.109

SDIF packer

A program that creates an SDIF data stream.

3.110

SDIF unpacker

A program that decomposes an SDIF data stream into its constituent entities.

NOTE 59 If necessary, the SDIF unpacker will modify the system identifier parameter of markup declarations to be consistent with storage addresses in its environment.

3.111

self anchor

A hyperlink that addresses itself as one or more of its anchors.

3.112

semantic grove

A grove created by an application or architecture engine. The nodes of a semantic grove reflect the data structures required to implement the application- or architecture-specific semantics.

NOTE 60 The HyTime architecture's semantic grove is defined by the HyTime property set (see annex B).

3.113
SGML Document Interchange Format
SDIF
A data structure that enables a main document and its related documents, each of which might be stored in several entities, to be combined into a single data stream for interchange in a manner that will permit the recipient to reconstitute the separate entities.

NOTE 61 When SDIF is used as a hyperdocument interchange format for HyTime, the “main” document, if there is more than one, is the hub document.

3.114
SGML processing link
The “link process” feature of SGML; it allows multiple context-sensitive sets of presentation and processing specifications to be associated with elements of a document.

3.115
SGML subdocument entity
SUBDOC
An SGML entity that conforms to the SGML declaration of the SGML document entity, while conforming to its own document type and link type declarations. It contains, at a minimum, a base document type declaration and the start and end of a base document element.

3.116
subhub
A hub document declared in another hub document and whose own HyTime BOS is added to the parent document's HyTime BOS.

3.117
SMU name
A name, declared by a notation declaration, that identifies a standard measurement unit.

3.118
(location) step
An element in a location path.

3.119
subnode
In groves, a node whose origin node exhibits a property of which the node is a member.

3.120
subnode property
In groves, a property whose value must consist of nodes whose origin is the node exhibiting the property.

3.121
subnode tree
The tree of nodes consisting of a node and all of its subnodes.

3.122
target (of an ID reference)
The referent or, if that is a location address, the objects located by the location path that the referent begins.

3.123
treeloc
Tree location address.

3.124
tree location address
A location address that addresses a single node of a tree in the classical manner by selecting a node from an addressable range at each level of the tree, starting at the root.

3.125
validating HyTime engine
A conforming HyTime engine that can find and report a reportable HyTime error if (and only if) one exists.

3.126
view
A presentation of a web and the anchors that it connects.

3.127
web
A set of one or more hyperlinks that are used together.

NOTE 62 Typically, the hyperlinks deal with a common topic and/or they can be traversed continuously through common anchors and/or anchors occurring in the same object.

4 Symbols and Abbreviations

The following abbreviations are used in this International Standard.

BOS	Bounded object set.
DTD	Document type definition.
FCS	Finite coordinate space.
GMT	Greenwich Mean Time.
HMU	HyTime measurement unit.
HyTime	Hypermedia/Time-based Structuring Language.
IOH	Integrated open hypermedia.
JD	Julian date.
LPD	Link process definition.
MDU	Measurement domain unit.
RHE	Reportable HyTime error.
SDIF	SGML Document Interchange Format.
SGML	Standard Generalized Markup Language.
SMU	Standard measurement unit.
SUBDOC	SGML subdocument entity.
UTC	Coordinated Universal Time.

5 Notation

HyTime is an enabling document architecture whose definition conforms to the Architectural Form Definition Requirements in annex A.3 of this International Standard.

NOTE 63 That annex should be read in conjunction with this clause.

The specification of HyTime is accomplished by a combination of narrative text and formal definitions.[1] The formal definitions are expressed in SGML.[2]

NOTE 64 This document uses editorial conventions mandated by the ISO with which the reader should be familiar in order to understand the implications of certain words. Sources for a brief informal explanation of these conventions, and of the subset of SGML that is used in this document, can be found in annex D. However, the specific conventions for using SGML to define HyTime architectural forms are described in annex A.3.

The text describing each construct emphasizes semantics, while the formal SGML definition provides the rigorous syntactic definitions underlying the text descriptions. The text occasionally discusses syntactic issues that require explanation beyond that provided in the formal SGML, but normally such issues as the syntactic constraints on an attribute value, or the default value supplied by HyTime, are not described in the text.

NOTE 65 For this reason, it is recommended that the reader refer to the SGML definitions while reading the textual descriptions. Although the SGML definition always follows the related text, the user may find it helpful to read the SGML first in some cases. Complete meta-DTDs are found in annex C.

When a construct is first introduced, it is described in the text. If the construct occurs in the formal SGML specification, both the formal SGML name and a full name in English are presented, as follows:

> The element form **full construct name** (*SGMLname*) ...
>
> The attribute form *SGMLname* ...
>
> The attribute **full construct name** (*SGMLname*) ...

If a construct occurs in several places (as in the case of the same attribute with several element forms) the description is not repeated.

The declarations include comments, called “conventional comments”, that follow conventions established in this clause to specify syntactic and semantic constraints and other information that is known to the HyTime engine. The conventional comments do not extend SGML in any way. They are used in the architecture definitions only, as a notation for the documentation of the architecture. They need not be included in application DTDs and, if they are included, the SGML parser will treat them as it would any other comment.

5.1 RCS name, full name, description, and clause

Every form name is followed by comments giving the full name, a description of the form, and the number of the clause in this International Standard in which the form is defined.

[1] This document makes several references to industry and proprietary standards, products, user groups, and publications. Such references are not normative, and do not imply endorsement by ISO, IEC, or their national member bodies or affiliates. Any brand names or trademarks mentioned are the property of their respective owners.

[2] The formal SGML definitions are part of the text of this International Standard and are protected by copyright. In order to facilitate conformance to HyTime, the formal SGML definitions may be copied as specified in the following copyright notice: *(C) International Organization for Standardization 1992, 1997. Permission to copy in any form is granted for use with conforming HyTime systems and applications as defined in ISO/IEC 10744, provided this notice is included in all copies.* The permission to copy does not apply to any other material in this International Standard.

Attribute forms that are used by more than one element or notation form are given names in the reference concrete syntax for ease of reference. The RCS name is given in a comment immediately following the ATTLIST keyword of the attribute list declaration. Attribute lists that apply to a single element or notation form do not have this comment.

Individual attributes have a full name and description comment.

5.2 Lexical type

The lexical types of attributes and data content are specified by conventional comments beginning with the word “Lextype” or “Ulextype”, followed by a lexical model in the “HyTime lexical model notation” (HyLex) (see *A.2.3 HyTime lexical model notation (HyLex)*). The word “Lextype” signifies a normalized lexical model, while “Ulextype” signifies an unnormalized one. The lexical type names used in conventional comments are defined in the HyTime lexical type set (see *C.1 HyTime Lexical Types*).

Names matching ATTNAME tokens in lexical types are searched for first in the client element or entity, followed by each architectural element or entity in the chain from the client element or entity to and including the architectural element or entity to which the ATTNAME lextype applies.

Names matching NOTATION tokens in lexical types are searched for first in the client document, followed by each architectural document in the chain from the client document to and including the architectural document to which the NOTATION lextype applies.

NOTE 66 HyLex is not only used to document architectures. It is part of the General Architecture and can therefore be used by application designers as well. However, use of HyLex in conventional comments in this International Standard does not require that implementations support its use by applications.

5.3 Constraints

Comments labeled “Constraints” define additional semantic or syntactic constraints on the constructs they follow. Constraint comments that follow the name of a form define constraints on the use of that form in general. Constraint comments that follow a component of a declaration (for example, the default value prescription of an attribute declaration), define the specific constraints on that component.

5.4 Note

Note comments provide additional information not provided by the other comment types and are informational rather than constraining.

5.5 Associated attribute forms and attribute lists

Within its declaration, an element form or notation form identifies its associated attribute forms by means of conventional comments consisting of any of the words “Attributes”, “OptionalAttributes”, or “CommonAttributes”; followed by the names of the attribute forms.

The “Attributes” comment lists attribute forms that are always associated with the form. The “OptionalAttributes” comment lists attribute forms that are associated with optional facilities that are not themselves required by the form; these attribute forms will only be provided by the form when the relevant optional facilities are supported. The “CommonAttributes” comment lists the common attribute forms of the architecture as a reminder of their availability.

5.6 Referrers

The “Referrers” comment lists those element forms that may or are required to refer to this form. The comment is a list of element forms, followed by the name of the attribute by which the reference is made unless the reference is made in element content.

5.7 Conventions for attribute form declarations

Declared values that require ID references are specified with either the SGML IDREF or IDREFS keywords, or with ID references indicated by a lextype conventional comment. All attributes that are semantically referential, regardless of declared value, are so indicated by the use of a conventional comment consisting of the word “Reference”.

If a referential attribute definition contains a conventional comment containing the word “Reftype” followed by a name or SGML OR group, the target or targets of the reference must be elements that conform to the named element form or an element form whose name occurs in the OR group. Note that the OR group is meta: the names are architectural forms rather than element types.

NOTE 67 The reftype conventional comment should not be confused with the more powerful reftype facilities of the General Architecture and HyTime, which are for use by application designers. Use of the reftype comment in an architecture definition does not require the architecture to support the General Architecture or HyTime reftype facilities.

An attribute's default value is documented in the default value prescription of the attribute definition, except that for impliable and content-reference attributes it is documented in a conventional comment that begins with “Default”.

If the attribute definition contains a conventional comment that consists of the word “Constant”, it signifies that the attribute is a constant attribute; that is, the attribute must have the same value for all elements of a given element type if it has a value for any element of that type.

NOTE 68 In SGML, a fixed attribute is one whose specified value (if any) must always be equal to its default. A fixed attribute is necessarily a constant attribute, but the converse is not necessarily true.

NOTE 69 If an attribute is constant in the meta-DTD, an application should declare it as a fixed attribute in the DTD, but this is not required because an architecture only defines conformance for document instances. However, a validating HyTime or General Architecture engine can optionally warn when this is not the case.

The “Constant”, “Lextype”, “Ulextype”, “Reference”, and “Reftype” conventional comments could be recognized and processed to provide validation beyond that supported by SGML alone.

5.8 Identification of optional facilities

HyTime's facilities are organized into modules, each of which is documented in a separate clause of this International Standard. All modules except the base module are optional and most facilities of each module are optional. Whether or not a facility is optional is indicated by the use of parameter entities to control marked sections in the meta-DTD. For each optional facility there is a parameter entity whose name is the same as the name of the optional facility. These parameter entities are set to “INCLUDE” by naming them in one of the architectural facility control attributes defined for the HyTime notation (see *6.3 HyTime support declarations*).

The total set of options is summarized in the integrated HyTime meta-DTD provided in *C (normative) Architectural Meta-Declarations*.

6 Base module

This clause describes the HyTime base module, which is required for all uses of HyTime.

6.1 Concepts and definitions

Some key concepts relating to the facilities of the base module are described in this sub-clause.

6.1.1 Object representation

SGML is used for source object representation in a HyTime document. Groves are used for the representation of abstract objects derived from source objects.

NOTE 70 While the source of HyTime documents is represented in SGML, all HyTime semantics are defined in terms of operations on nodes in groves, e.g., on the abstract data objects described by SGML documents (or any other type of data for which a suitable grove representation can be provided).

This sub-clause describes the key SGML constructs on which HyTime relies.

NOTE 71 Components of a HyTime hyperdocument that are not HyTime documents need not be represented in SGML. A document that is used as a hub document of a HyTime hyperdocument must be a HyTime document, and therefore must be represented in SGML.

SGML is a conceptual tool for the modeling of information structures intended for human perception (called “documents”), as well as a notation for representing them. Documents are analyzed and represented in terms of three main constructs:

element A structural building block that can be defined to contain data, subordinate elements, or both. In typical documents, examples of elements would be paragraphs, chapters, headings, figures, and tables. In hypermedia documents, some examples are hyperlinks of various types, event schedules, events, and objects encoded in a variety of data notations.

attribute A property, associated with elements of a given type, whose value describes the element but is not part of its content. For example, the owner of a chapter or revision date of a table. Many element types have an *ID* attribute, which provides a label for each element of that type so that it can be referenced explicitly from any place in the document. The references are made by elements that have an *ID reference* attribute whose value is the label of the element referenced (as defined by that element's ID attribute).

entity A unit of virtual information storage that contains part of a document (sometimes all of one or even more than one). An entity can be referenced by a name from one or more places in a document, thereby causing its information to be included in the document at the points of reference (see *6.1.1.1 Entity structure*).

Entities are independent of elements. For example, an entity could contain part of a picture element or one and one-half paragraph elements.

The particular instances of those constructs that are permitted in a given document are declared in a “document type definition” (DTD) to which the document conforms. The designer of a DTD can choose to allow parts of it to be extended or modified (typically by redefining entities that are referenced in the DTD). This technique provides the flexibility necessary to accommodate a wide variety of applications.

Processors that act on the semantic objects and their data content do not operate directly on the source representation. Instead, they must first create an abstract, “in-memory” representation of the structures and data in the source. In the case of SGML, an SGML parser processes the source document to recognize markup and data and provides information about what it found to a processing application. The processing application then operates on its particular abstraction derived from the result of parsing the document.

The HyTime standard uses a standard form of abstract structure representation called “groves”. All HyTime-defined processing operates on groves (rather than on the unparsed source data from which groves are constructed). Groves consist of “nodes”, which exhibit “properties”. Each node is of a particular node class. The node classes and their properties are defined in “property sets”.

The combination of property sets and groves provides a standard for the representation of application-specific data views of SGML documents (and, potentially, other data notations) and enables interoperation of applications by, at a minimum, providing a common design formalism by which applications can define the data objects on which they operate and the results of those operations, irrespective of how they actually represent or process data objects internally. In other words, groves and property sets make it possible to discuss the processing of structured data without first defining the implementation of the processors themselves. Groves are discussed in detail in *7.1.4 Groves and Location Addressing* and in *A.4 Property Set Definition Requirements (PSDR)*.

6.1.1.1 Entity structure

SGML includes a virtual storage model called the “entity structure” which allows a user to divide a document arbitrarily for ease of management. The entity structure is “virtual” because the mapping from entities to real storage is implementation-dependent, and there need not be a one-to-one relationship between entities and storage objects.

The entity structure is not typically reflected in the part of the document type definition that defines the structure that the application processing is concerned with — the “element structure”. The independence of the entity structure is one of the great strengths of SGML, and is essential for hypertext. For example, it relieves an application designer from the burden of having to predict whether chapters will span multiple storage objects or whether a single storage object will contain multiple chapters.

6.1.1.2 Data

The term “data” is used to distinguish the information in a document that is not part of the document structure. For example, the character text within a paragraph, or the raster information representing a photograph, is data. An external entity containing a document that uses a different document representation from the document referencing it is also data.

NOTE 72 Similarly, a recursive instance of an SGML document or subdocument is data with respect to the current instance, because, as it has its own DTD, it is not parsed in the current SGML parsing context.

Data is sometimes referred to informally as “content”, but this usage is ambiguous because the content of an element is not restricted to data; it could also be subelements exclusively, or mixed subelements and data. Data can also occur outside the content of an element in attribute values or external data entities.

NOTE 73 In SGML, data is also defined as “that which is not markup”, but the two definitions are consistent: In an SGML document, information that is considered data because it is not structural will also not be markup, and vice versa.

HyTime provides facilities for addressing all forms of data, either in terms of HyTime constructs, or by interfacing to “location addresses” that understand the data representation (see clause 7).

6.1.2 Object identification and addressing

Hypertext linking and multimedia time synchronization are applications of the same basic function — addressing.

A hyperlink may address its anchors by names that are unique within some name space or by position within a list or tree. Time synchronization uses coordinate addresses of events on a time axis: the address of event A may be expressed in terms of its position and size on the axis, or in terms of its relationship to the address of event B. Spatial alignment is similar, except that the axes are measured in spatial units, rather than temporal.

In HyTime, addressing is applied to nodes in groves and the result of any location address is the list of nodes addressed. Syntactically, the list of nodes addressed can be given a unique name by specifying a unique ID for the location address element that addresses the nodes. Whether it has a name or not, every such node list is uniquely addressable because it becomes the property of a node in the “HyTime semantic grove”. The nodes exhibiting these properties can then be addressed by name, if they have one, or by other node addressing methods, if supported.

NOTE 74 If an application only supports ID-based addressing, all elements on which other elements depend and use by reference must have IDs.

6.1.2.1 Name space addressing

In the grove model, any node class may exhibit a “name” property that uniquely identifies it within a “name space”. In groves, a name space is a property whose value is a list of nodes, all of which share the same “name” property and each of which exhibits a unique value for its name. SGML defines two primary name spaces that are made directly accessible through the syntax of SGML documents: application-defined information components, called elements, and units of storage segmentation of information, called entities.

By and large, applications deal only with elements, while storage is managed transparently. Therefore, the fundamental form of name is one that is unique among the elements of a document, known as a unique identifier (ID). The fundamental means of specifying an object to reference is called an “ID reference” (IDREF). Like the ID, it is an attribute of an element.

Conceptually, all location addresses begin as ID references.

NOTE 75 HyTime also provides “shortcut” forms of address in which the ID reference or references are implicit. For example, when coordinate addressing is supported, the reftype facility allows the direct use of coordinate addresses in place of ID references. However, for all such shortcuts, the equivalent reference can be constructed using ID references and other fundamental forms of addressing.

For locations outside the current document, HyTime uses a standardized system of identification, including public and private, local and global, unique identifiers (see ISO 9070) in order to address the entities that contain those locations.

6.1.2.2 Coordinate addressing

A coordinate system in HyTime is called a “finite coordinate space”. It consists of a set of coordinate axes and a system for measuring along them.

Each axis is treated as an ordered set of “quanta”. A coordinate address consists of a position (the first quantum of interest) and a specific number of subsequent contiguous quanta for each of the axes of the coordinate space. This combination of position and size is called an extent.

When the scheduling module is supported, occurrences of objects (“events”) can be given extents in coordinate spaces. Events can be aligned with one another by defining their extents with reference to the extents of other events.

When the location address module is supported, “location address” elements can be defined that associate an ID (directly or indirectly) with a coordinate address. This type of location address allows references to be made to objects that can be identified only by their position.

NOTE 76 For example, “the third word in the sentence”.

It also allows references to arbitrary portions of an object.

NOTE 77 For example, “the second and third characters of the third word in the sentence”.

6.1.2.3 Semantic addressing

Any object, in any notation, can be represented in a HyTime hyperdocument. The object as a whole could be an element, and therefore could have an ID. The object could be included in an event, and thereby have a coordinate address.

When the location address module is used, it is also possible to address an object by a query against its properties. Nodes in any grove, regardless of the data notation from which they were derived, are addressable by HyTime location addresses. Because the properties of nodes in groves are formally defined in a property set definition, it is possible to have general query facilities that can operate on any grove. Groves can, potentially, be constructed from any data notation by a "grove construction process" that understands the syntax and semantics of the notation (however the degree to which a grove can fully express the structure or content of a given notation is dependent on its similarity to SGML — groves are not intended nor guaranteed to be capable of fully representing any possible notation).

NOTE 78 In practical terms, "constructing a grove" may simply mean providing an interface that takes as input a HyTime location address and returns the data that would have been addressed had a literal grove been constructed. In other words, there is no requirement to literally build a grove; it is only necessary to behave as if one had been built.

6.2 Hyperdocument management facilities

Hyperdocument management facilities are required for all uses of HyTime. They include the representation, identification, and accessing of objects, and a hyperdocument interchange format.

6.2.1 Object representation

Representation facilities for HyTime documents are provided by SGML and are defined in ISO 8879. HyTime requires the use of the SGML formal public identifier feature, but does not otherwise constrain the choice of available SGML options and variants.

The document representation facilities include:

— Document type definition capability, including user extensions that can be controlled by the application designer.

— Representation of any architectural structure.

— Demarcation of information objects, including components of documents, such as elements, attributes, and data content portions.

— Ability to include multimedia object formats represented in any notation.

— Separation of processing specifications from intrinsic information.

— Ability to associate multiple sets of processing specifications ("styles") with element types and individual elements.

— Independent transparent entity structure (see *6.1.1.1 Entity structure*).

NOTE 79 Within a HyTime hyperdocument, documents and information objects other than the hub document need not be HyTime documents and need not be represented in SGML. For objects that have an internal structure, those structures will only be addressable through HyTime facilities if a grove is constructed for them. An object for which there is no grove can only be addressed by reference to an entity node in an SGML document grove that declares the object as a data entity.

6.2.2 Object identification and addressing

Object identification facilities within HyTime documents are provided by SGML and are defined in ISO 8879. Public identifiers of external documents and information objects must conform to ISO 9070.

NOTE 80 ISO 9070 allows public identifiers to be represented in a variety of formats, including ASN.1 structures and SGML formal public identifiers. The identified objects need not be represented in SGML.

NOTE 81 Forms of public identifier not supported by ISO 9070, such as distributed object references with semantic connotations, can be treated as formal system identifiers (see *A.6 Formal System Identifier Definition Requirements (FSIDR)*).

6.2.3 Object access

Access from the identification of an object to its physical storage is provided by the SGML entity manager component of a HyTime system, which might invoke network processes to retrieve the located data from a remote data base.

NOTE 82 The process is analogous to the role of a reference librarian, who accesses a document mentioned in a citation either by locating it on the library shelves, or by consulting a catalog to find it in another library.

6.2.4 Bounded object set (BOS)

Several HyTime semantics, including hyperlink anchor semantics, activity policy semantics, and scheduling semantics, are applied to referents by referencing constructs (referrers). The referent of any referrer may be in a different object than the referrer. Since no system can be expected to process all of the documents in existence to determine what semantics may have been applied to any given piece of information, it is necessary to specify which documents contain all the referrers that are needed in order to understand the referents in a given HyTime hyperdocument.

Also, as a HyTime hyperdocument may consist of more than one object, authors must be able to specify what objects need to be present in order to have a complete copy of the hyperdocument.

For these reasons, the number of documents and other information objects that a HyTime system can be expected to regard as a hyperdocument must be finite — a “bounded object set” (BOS). If the exact set of required objects is not known, there is a potential for incorporating vast amounts of unwanted information and incurring unnecessary delay, expense and inconvenience. HyTime provides facilities for managing bounded object sets. These facilities consist of common data attributes, attributes of the HyDoc element form, and the BOS specification element form (see *6.5.2 HyTime BOS control data attributes* and *6.4 HyTime document*).

Conceptually speaking, there are three kinds of BOS:

HyTime BOS Every HyTime hyperdocument has exactly one “hub” document; processing of the entire HyTime hyperdocument nominally begins with the hub, which is nominally the HyTime document through which entry to the HyTime hyperdocument was gained, that is, the HyTime document with which the processing session begins. The HyTime BOS is the BOS specified (directly and indirectly) by a HyTime hub document. Therefore, the author of a HyTime hyperdocument has control of the set of objects considered to be members of its HyTime BOS. The membership of objects in the HyTime BOS is determined according to rules defined by this international standard regarding the semantics of the boslevel, inbos, and subhub common data attributes, the semantics of the maxbos, boslevel, and bosspec attributes of the HyDoc architectural form, and the semantics of the boslevel and inbos attributes of the bosspec architectural form. The objects in a HyTime BOS are also classifiable by HyTime hyperdocument authors as having foreground or background priority.

application BOS

The application BOS is the BOS intended by the software programmer of a HyTime system to constitute the hyperdocument.

NOTE 83 The concept of the application BOS includes an implicit recognition of the fact that some systems may be deliberately designed to process hyperdocuments which differ, in terms of the set of member objects used, from the HyTime BOS. For example, there may be systems that regard all documents referenced by World Wide Web-style URLs that appear in the hub document to be in the BOS, even when the documents referenced by such URLs are not specified by any corresponding entity declarations anywhere in the hyperdocument.

NOTE 84 If no value is specified for the boslevel attribute of the hub document's HyDoc document element, there is no HyTime BOS and, of necessity, the application BOS governs the assembly of the effective BOS.

effective BOS

The BOS consisting of all the objects that at any given point have been successfully and fully integrated into the hyperdocument being processed.

NOTE 85 The concept of the effective BOS is an implicit recognition of the fact that some objects that the hyperdocument author regarded as necessary (and which are therefore in the HyTime BOS), and/or which the system programmer regarded as necessary (and which are therefore in the application BOS), may not be retrievable or integratable into the hyperdocument with complete success. Much of this international standard is primarily concerned with the semantics of a set of objects that has been fully and successfully integrated into a hyperdocument, and therefore it is primarily concerned with the effective BOS, irrespective of how the membership in the effective BOS was determined, or which members of the HyTime and/or application BOSs had to be left out or exceptionally included for any reason.

As the HyTime BOS is the only BOS whose membership is defined by this international standard, and as the HyTime BOS is the only standardized indication of what objects an author intends the hyperdocument to consist, there is reason to expect that, in many applications and instances in which HyTime hyperdocuments are processed, the application BOS and the HyTime BOS will be identical, and, therefore, barring mishap, the effective BOS will be the HyTime BOS.

This international standard does not provide means for expressing application BOSs or effective BOSs; it provides such means only for HyTime BOSs.

NOTE 86 Of course, nothing prevents the expression of an application BOS or an actual effective BOS as a HyTime BOS in a HyTime document.

Except possibly for addressing information needed to refer to portions of the content of documents that are not present in the effective BOS, a HyTime system is not required to have any knowledge of the content of documents and objects that are not in the effective BOS. If a HyTime system obtains access to the content of such documents, such as when resolving the addresses of anchors within such documents, such documents must first be added to the effective BOS. Insofar as HyTime processing semantics and the system's requirements dictate, a HyTime system maintains knowledge of and access to all the referents within the effective BOS of all referrers within the effective BOS.

NOTE 87 In other words, the effective BOS is the set of information objects for which a HyTime system has actually assumed responsibility at any point in time. The effective BOS concept — the idea that a system has only a finite number of objects for which it is responsible — makes references with bidirectional semantics from and to external documents both practical and scalable.

For example, if the scheduling module is supported, the effective BOS contains all the fcs elements corresponding to all the FCSs for which the system is responsible, and all the scheduling constructs known to affect them. Even if an fcs element exists in the effective BOS, any schedules, batons, or wands that are directly scheduled or projected onto the logical FCS to which it corresponds will have no effect unless they, too, are included in the effective BOS. (In other words, the system is not required to know about anything affecting FCSs that is not in the effective BOS.) Schedules, batons, and wands that only affect FCSs that correspond to fcs elements that are not present in the effective BOS have no effect unless and until the documents in which such fcs elements are declared are added to the effective BOS. (Systems may support the addition of documents containing such referenced fcs elements to the effective BOS even when they are not members of the HyTime BOS; such added objects are, by definition, members of the application BOS.)

For another example, if the activity option of the base module is supported, the effective BOS determines the effectivity of activity policies. Actrules in the effective BOS are in effect for the objects in the effective BOS that they govern, according to the policies that are also in the effective BOS.

For a third example, in the case of hyperlinks, the system knows all the anchors within the effective BOS of all the links within the effective BOS. It need not know the anchors of any links if those anchors are not in the effective BOS, and it need not know any anchors that are the anchors of links that are not in the effective BOS. If a link is traversed to an anchor outside the effective BOS, the object containing that anchor must first be added to the effective BOS. Some systems may do this without any special user interaction; others may warn users that, for example, the effective BOS will thereafter contain an object that was not specified in the HyTime BOS.

NOTE 88 Adding a document to the effective BOS does not necessarily mean adding that document's HyTime BOS to the effective BOS, as if the added document were itself the hub document. Only the hub document specifies how the HyTime BOS is to be derived, and, in any case, some systems may ignore even the HyTime BOS.

A HyTime bounded object set (“HyTime BOS”) can be determined automatically by a HyTime system by “discovering” an “entity tree”, using the SGML document entity of the hub document as the root. The entity tree includes external document, subdocument, and data entities declared in the hub document, then external entities declared in those entities, and so on. Only entities declared in “external identifier” parameters of markup declarations will be discovered. Therefore, if external references are made from data content using a notation-specific form of reference, such entities will not be included in the HyTime bounded object set unless the document originator also creates entity declarations for them.

NOTE 89 This international standard does not require systems to assemble BOSs according to any particular methodology or order. However, purely as an example, it may be useful to consider the following methodology for determining the membership of objects in a HyTime BOS. (Considerable optimization of the performance of interactive HyTime hyperdocument applications can be achieved by judicious use of foreground and background priority. For clarity, the following methodology does not take the possibility of background priority into account.)

1) Determine the unadjusted HyTime BOS.

   a) For each entity declared in the hub document entity whose subhub attribute's value is subhub, complete a HyTime BOS assembly process exactly like the one being done for the hub document. This should be done recursively if subhubs declare entities as subhubs. The HyTime BOS of each subhub document entity is added to the tentative HyTime BOS of its parent document entity. Do not regard as a subhub any entity whose parent is not a hub or subhub, even if its parent declares it using a subhub attribute with a value of “subhub”.

   b) Discover the entity tree of which the SGML hub (or subhub) document entity is the root, adding each entity to the tentative HyTime BOS. Ignore all entities whose subhub attribute's value is “subhub” unless their parent entity was neither a hub nor a subhub, as they have already been taken into account in the previous step. During the entity tree discovery process, do not exceed the number of recursions specified by the smaller of (i) the value of the maxbos attribute of the hub's (or subhub's) document element's maxbos attribute, or (ii) the value of the boslevel common data attribute (which may have a default value specified by the boslevel attribute of the hub's [or subhub's] document element's boslevel attribute) of the corresponding entity declaration in the hub document entity. If, during the entity tree discovery process, an entity declaration's inbos attribute has a value of “inbos”, include it, too, in the tentative HyTime BOS even if doing so exceeds the maxbos or relevant boslevel specification.

2) Apply all adjustments specified by bosspecs referenced by the bosspec attribute of the document element of the hub (or subhub) document whose inbos attribute's value, if any, is not “notinbos”. This may require additional entity tree discovery processing. If the value of the inbos attribute of a bosspec is “inbos”, add all the objects addressed by the content and boslevel attribute of the bosspec to the tentative HyTime BOS.

3) Apply all adjustments specified by bosspecs referenced by the bosspec attribute of the document element of the hub (or subhub) document whose inbos attribute's value is “notinbos”. Whatever objects remain constitute the HyTime BOS.

Limits can be placed on the depth of the HyTime BOS entity tree discovery process by the maxbos attribute of the hub document's document element, by the boslevel common data attribute on entity declarations, and by bosspec elements referenced by the bosspec attribute of the hub document's document element.

The HyTime BOS, if specified, represents the author's intentions with respect to the objects that participate in the hyperdocument. However, a system may use either an application BOS or a HyTime BOS to guide the assembly of its effective BOS. Moreover, it is possible for systems to exclude, voluntarily or involuntarily, some of the member objects of the HyTime BOS from the effective BOS, and/or to include objects in the effective BOS that are not members of the HyTime BOS. One possibility is for the system to use the HyTime BOS as a starting effective BOS, but to allow entities to be added and deleted by the user. (By definition, once the effective BOS is no longer the same as the HyTime BOS, the effective BOS is an application BOS.) Therefore, the question of whether the system is presenting the hyperdocument as the author intended at any given point is an information accuracy issue.

It is possible for the effective BOS to be neither an application BOS nor a HyTime BOS, if the intended HyTime or application BOS cannot be fully assembled due to environmental processing constraints, such as the lack of a network connection. It is also possible for a system to regard a document other than the session-start document as a hub document, and to use the HyTime BOS that it specifies in some fashion.

All three types of BOS (HyTime, application, and effective) always contain, at minimum, a hub document.

6.2.5 Hyperdocument interchange format

A “hyperdocument interchange format” is a data structure that combines the separate documents and information objects of a hyperdocument into a single “data stream” for interchange. The hyperdocument interchange format recommended for use with HyTime is the SGML Document Interchange Format (SDIF) defined in ISO 9069.

NOTE 90 This International Standard does not constrain the interchange format for renditions of a HyTime hyperdocument, for which an application may choose to use a format optimized for transmission or interactive presentation.

SDIF is independent of the architectures and applications of the information objects that it carries. A single SDIF implementation could therefore be capable of supporting all HyTime applications.

An SDIF data stream can carry any bounded object set, and is not strictly limited to HyTime hyperdocuments. In particular, the interchange data stream could include all of a single hyperdocument, a part of it, and all or part of other documents as well. In all cases, SDIF preserves the demarcation of documents and information objects, and can preserve the entity structure in which they existed in the originator's environment.

6.2.5.1 SDIF packer

The process (application or HyTime engine) that determines the BOS submits the names of the entities that comprise the BOS to the SDIF packer program. The SGML document entity of the hub document, if present, is identified to SDIF as the “main” document. The packer creates an “entity descriptor” in the SDIF data structure for each entity. The descriptor includes the entity data. However, the SDIF packer will create a “cross-reference” entity descriptor if the external identifier of an entity duplicates that of another, in order to avoid duplicating the data.

NOTE 91 The SDIF “packer” process that creates the data stream is aware of the entity declarations, and therefore of the notations (media types) of the entities. In a communications environment it would be possible to use this information to optimize bandwidth allocation and routing for different types of media when transmitting the hyperdocument.

HyTime provides a means of including the data of several entities within the storage of a single “container” entity, thereby allowing the use of interleaving and other techniques that optimize access to multimedia data. The data of the container entity is stored in its entity descriptor, while the entity descriptors of the contained objects are cross-references to the container (see *A.6.3 Containers*).

6.2.5.2 SDIF unpacker

An SDIF unpacker receives an SDIF data stream and stores its contents in the storage media of the receiving system. Alternatively, it could act as a “handler” or “reader” service, invoked by the SGML entity manager, that unpacks SDIF dynamically as required by an application program.

If necessary, the SDIF unpacker modifies the system identifiers of the markup declarations to be consistent with storage addresses in its environment. When it encounters an entity declaration for which there is no entity descriptor (because, e.g., it is outside the bounds of a HyTime BOS), it modifies the system identifier to show that the entity is not available. An attempt by an application to access that entity will result in an appropriate message.

NOTE 92 An SDIF unpacker implementation could choose to retain the sender's original external identifier, together with a “not received” indication and the date and time of the SDIF receipt. Later, if an unsuccessful attempt is made to access the entity, the entity manager could give the application the option of contacting the sender to request a copy, or trying to access it directly on the sender's system.

6.3 HyTime support declarations

A document may indicate conformance to HyTime by following the procedures in *A.3.3.1 Enabling architecture use of APPINFO parameter*. That is, the APPINFO parameter of the SGML declaration declares the existence of an ArcBase declaration, which identifies the HyTime architecture support declarations.

NOTE 93 The use of APPINFO serves to declare the use of architectures and HyTime within SGML declarations. However, it is sufficient to enable architectural processing to declare the use of HyTime with the ArcBase processing instruction within DOCTYPE declarations.

The support declarations include support attributes that are defined for all architectures. They are discussed in *A.3.4 Architecture support declarations*.

The support attributes for HyTime include support attributes for each module that control the use of optional facilities that affect the syntax of the HyTime meta-DTD. Additional support attributes control the use of facilities that have semantic, but not syntactic, implications.

There is a facility support attribute named after each module. The attribute value contains the names of the module's optional facilities for which support is being declared. If only the module name is specified, then only support of the module's mandatory facilities is declared, if any (some modules have no mandatory facilities). If other facilities are specified then support for the module is implied and need not be specified. If the specified attribute value is the empty string (“”) or the attribute is not specified at all and the default value declared in the notation attribute declaration list is “#IMPLIED”, then the module is not declared at all.

NOTE 94 For example, the following attributes declare that support is required for the optional desctxt facility of the base module and the non-optional facilities of the sched module. The links and rendition modules are not required at all.

```
base="desctxt" sched="sched"
```

Some options require qualification in addition to stating their support. For these options, separate attributes are provided where the attribute name is the name of the option, for example:

```
manyaxes="3"
```

When an option qualification attribute sets limits, if no value is specified, there is no limit for that option. Option-specific attributes are ignored if the option to which they apply is not supported.

The attribute **highest quantum count limit** (*hyqcnt*) declares a limit that no quantum count in the document instance will exceed, including the FCS dimensions of coordinate axes and the number of nodes in node lists. The hyqcnt is specified as a power of two that exceeds the limit. The HyTime default (and the minimum for any use of HyTime) is 4,294,967,295.

NOTE 95 In other words, when the default hyqcnt is used, a HyTime implementation is never required to accumulate a quantum count greater than the capacity of an unsigned 32-bit integer: (2**32)-1.

An application designer is required to choose granularities that will not cause this limit to be exceeded.

NOTE 96 The hyqcnt gives the application designer wide latitude in specifying the precision of HyTime coordinate addressing. For this reason, HyTime's use of finite quanta rather than infinite points does not restrict HyTime applications in any practical way.

The attribute **any SGML declaration allowed** (*anysgml*) specifies whether (anysgml) or not (nanysgml) the HyTime processor allows members of a hyperdocument to use different SGML declarations. When the value is “nanysgml”, all SGML documents in the hyperdocument must conform to the same SGML declaration.

The attribute **external references allowed** (*exrefs*) specifies whether (exrefs) or not (nexrefs) external references are allowed. When external references are not allowed, all location addressing (except direct entity reference) must be to objects within the document from which the reference is made. The default is to allow external references. This option is ignored if the location address module is not supported.

The attribute **SGML model groups for reftype** (*refmodel*) specifies whether (SGMLmdl) or not (nSGMLmdl) reftype attributes may specify any valid SGML content model or are limited to repeating OR groups of element type names.

The attribute **maximum number of anchors allowed in hyperlinks** (*manyanch*) specifies the maximum number of anchors allowed for hyperlink instances. If the value is unspecified or is zero, the number of anchors is unlimited. This option is ignored if the hyperlinks module is not supported.

The attribute **maximum number of axes allowed in coordinate spaces** (*manyaxes*) specifies the maximum number of axes allowed for finite coordinate spaces. If the value is unspecified or is zero, the number of axes is unlimited. This option is ignored if the scheduling module is not supported.

The complete set of facility attributes, with all options supported, is:

```
                 <!-- HyTime Support Declarations -->
<?IS10744 ArcBase HyTime>
<!NOTATION
   HyTime         -- HyTime Architecture --
                  -- A base architecture used in conformance with the
                     Architectural Form Definition Requirements of
                     International Standard ISO/IEC 10744. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!ATTLIST #NOTATION HyTime
   ArcFormA NAME     HyTime
   ArcNamrA NAME     HyNames
   ArcSuprA NAME     sHyTime
   ArcIgnDA NAME     HyIgnD
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    "HyTime"

   -- NAMELEN must be at least 9 because the HyTime meta-DTD uses
      8-character parameter entity names. --
   ArcQuant CDATA    #FIXED "NAMELEN 9"

   ArcDataF NAME     #FIXED HyBridN
   ArcBridF NAME     #FIXED HyBrid

   -- For some HyTime forms, the generic identifiers of client elements
      are meaningful to a HyTime engine and must be customized.
      Therefore, automatic form mapping may be inappropriate. --
   ArcAuto  (ArcAuto|nArcAuto) nArcAuto

   ArcOptSA NAMES    "GenArc base links locs rend sched"
```

```
GenArc         -- General architecture facilities --
   CDATA       -- Lextype: csname+ --
   "altreps dafe dvlist HyLex HyOrd included ireftype lextype
    opacity REGEX superdcn"

base           -- Base module facilities --
   CDATA       -- Lextype: csname+ --
   "activity actypes bos bosspec conloc desctxt dimspec HyDimLst
    HyDimSpc HyFunk markfun valueref"

locs           -- Location address module facilities --
               -- Clause: 6.3 --
   CDATA       -- Lextype: csname+ --
   "agrovdef bibloc dataloc datatok grovplan listloc mixedloc
    multloc nameloc nmsploc pathloc pgrovdef proploc queryloc
    refctl referatt refloc reftype relloc spanloc treecom treeloc
    treetype"

links          -- Hyperlinks module facilities --
               -- Clause: 6.3 --
   CDATA       -- Lextype: csname+ --
   "agglink anchloc clink hylink ilink linkloc traverse varlink"

sched          -- Scheduling module facilities --
               -- Clause: 6.3 --
   CDATA       -- Lextype: csname+ --
   "calibrat calspec dimref fcsloc grpdex grprepet HyCalSpc
    HyExSpec HyExtLst HyGrand measure objalign pulsemap sched"

rend           -- Rendition module facilities --
               -- Clause: 6.3 --
   CDATA       -- Lextype: csname+ --
   "baton batonseq HyDimPro HyExPro HyPro modify obextent patch
    project proseq wand wndpatch"

anysgml        -- Any SGML declaration allowed --
               -- Clause: 6.3 --
   (anysgml|nanysgml)
   nanysgml

exrefs         -- External references allowed --
               -- Clause: 6.3 --
   (exrefs|nexrefs)
   exrefs

refmodel       -- SGML model groups for reftype --
               -- Clause: 6.3 --
   (SGMLmdl|nSGMLmdl)
   nSGMLmdl

hyqcnt         -- Highest quantum count limit --
               -- Clause: 6.3 --
   NUMBER      -- Constraint: power of 2 >= 32  --
   32

manyanch       -- Maximum number of anchors allowed in
```

```
                     hyperlinks --
                  -- Clause: 6.3 --
      NUMBER      -- Constraint: must be >= 2 --
      #IMPLIED    -- Default: no limit --

   manyaxes       -- Maximum number of axes allowed in coordinate
                     spaces --
                  -- Clause: 6.3 --
      NUMBER      -- Constraint: must be >= 1 --
      #IMPLIED    -- Default: no limit --
>
<!NOTATION AFDRMeta
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD Notation//EN"
>
<!ENTITY
   HyTime         -- HyTime meta-DTD --
                  -- Clause: 6.3 --

   PUBLIC "ISO/IEC 10744:1997//DTD AFDR Meta-DTD
           Hypermedia/Time-based Structuring Language (HyTime)//EN"

   CDATA AFDRMeta
>
```

6.4 HyTime document

The element form **HyTime document** (*HyDoc*) is an architectural document element form (ArcDocF) (see *A.3.6.1 Architectural document element*).

```
              <!-- Content model parameter entities -->
<!entity %
   HyCFC          -- HyTime context-free content --
                  -- Note: %loc, %link, %resorce qualify but are used
                     as meta-inclusions rather than meta-proper-
                     subelements --

   "%GACFC;|HyBrid|fcs"
>
<!entity %
   loc            -- Location address forms --

   "anchloc|bibloc|dataloc|fcsloc|linkloc|listloc|mixedloc|nameloc|
    nmsploc|pathloc|proploc|queryloc|relloc|treeloc"
>
<!entity %
   link           -- Hyperlink forms --

   "agglink|clink|hylink|ilink|varlink"
>
<!entity %
   resbase        -- Base module resource forms --

   "actrule|bosspec|desctab"
>
<!entity %
```

```
   resloc         -- Location address module resource forms --

   "agrovdef|datatok|grovplan|pgrovdef"
>
<!entity %
   resschd        -- Scheduling module resource forms --

   "calspec|evsched|extent|extlist|measure"
>
<!entity %
   resrend        -- Rendition module resource forms --

   "baton|batrule|batseq|modpatch|modrule|projectr|prorule|proseq|
    rendrule|wandrule|wand|wndpatch"
>
<!entity %
   resorce        -- All resource architectural forms --

   "%resbase;|%resloc;|%resschd;|%resrend;"
>



<!element
   HyDoc          -- HyTime document element --
                  -- Clause: 6.4 --
   - O
   (%HyCFC;)*
   +(%link;|%loc;|%resorce;)

-- OptionalAttributes [base]: bos, bosspcat --
-- OptionalAttributes [locs]: dgrvplan --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
```

6.5 HyTime Bounded object set

Support for the management of HyTime bounded object sets is managed through the HyTime bos and bosspcat attribute forms of the HyDoc element form, the bosdatt data attribute form, and the bosspec element form.

6.5.1 HyTime bounded object set attributes

The attributes of the **HyTime bounded object set** (*bos*) attribute form control the construction of HyTime bounded object sets when the HyDoc elements that exhibit them are the document elements of HyTime hub documents.

Conceptually speaking, when determining which objects are members of the HyTime BOS of a HyTime hyperdocument, a HyTime application uses an “entity tree discovery” process in which entities that are recursively declared are examined and reported as being candidates for membership in the HyTime BOS (see *6.2.4 Bounded object set (BOS)*). The attribute **maximum bounded object set level** (*maxbos*) specifies the nominal maximum number of recursions permitted in the normal entity tree discovery process. The value of the maxbos attribute applies to all entities declared in the hub document except subhub document entities (see *6.5.2 HyTime BOS control data attributes*).

NOTE 97 The value of maxbos is said to specify the “nominal” number of recursions because it may be overridden by the inbos common data attribute, by the inclusion of subhubs, and by the bosspec attribute of the HyDoc element.

A maxbos value of 0 means that the number of recursions to be made in the course of the normal entity tree discovery process is not limited. A maxbos value of 1 means that only the hub document has nominal membership in the HyTime BOS.

NOTE 98 In other words, if maxbos is 1, the normal entity tree discovery process does not recurse at all.

A maxbos value of 2 means that the normal entity tree discovery process should stop after one recursion, that is, all of the entities declared in the hub document are nominally members of the HyTime BOS, but there should be no examination of the entities declared in the entities declared in the hub document. A maxbos value of 3 means there is a limit of two recursions, and so on.

The maxbos value takes precedence over the boslevel attribute of a data entity when the data entity's boslevel would result in a BOS larger than that allowed by the maxbos level.

NOTE 99 In other words, the nominal maximum number of recursions in the normal entity tree discovery process whose root is an entity declared in the hub document is actually limited to the smaller of the value of the maxbos attribute of the hub document's HyDoc element and the value of the boslevel common data attribute of such entity.

The attribute **bounded object set level** (*boslevel*) specifies the default bounding level to be used for data entities that do not specify a boslevel data attribute. The semantics of the numbers used in its value are the same as those of the maxbos attribute. If no value is specified for the boslevel attribute, there is no HyTime BOS; the author of the hyperdocument is, in effect, delegating responsibility to the system and asking that the application BOS govern the assembly of the effective BOS (see *6.2.4 Bounded object set (BOS)*).

```
<![ %bos; [
<!attlist
   -- bos --      -- HyTime bounded object set --
                  -- Clause: 6.5.1 --
  (HyDoc)

   maxbos         -- Maximum bounded object set level --
                  -- Bounding level of HyTime bounded object set when
                     document is a hub or subhub. --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=hub only) --
      0

   boslevel       -- Bounded object set level --
                  -- Default BOS level used by data entities declared
                     in hub document. --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=this entity only) --
      #IMPLIED    -- Default: No HyTime BOS --

>
]]><!-- bos -->
```

6.5.2 HyTime BOS control data attributes

The attribute form **HyTime BOS control data attributes** (*bosdatt*) consists of common data attributes used to control entities' inclusion in the HyTime bounded object set. The HyTime common data attributes can be used with any data content notation.

The attribute **BOS level** (*boslevel*) specifies the depth of entity reference relative to this entity to include in the bounded object set (see *6.2.4 Bounded object set (BOS)*). If not specified, the BOS level specified for the HyTime document element is used. The boslevel value is subordinate to the maximum BOS defined for the hub or subhub document. The boslevel value cannot cause entities to be included in the BOS if doing so would exceed the maximum BOS level.

The attribute **include in BOS** (*inbos*) indicates whether the entity must be unconditionally included in the BOS ("inbos") or is to be unconditionally excluded from the BOS ("notinbos"). If no value is specified, the entity is included when allowed by the governing BOS level. When "inbos" is specified, the entity will be included in the BOS unconditionally. Entities declared by an entity so included are included in the BOS only if they do not exceed the maximum BOS level or if they are also declared with a value of "inbos".

NOTE 100 Entities "unconditionally" included or excluded are still subject to inclusion or exclusion by bosspecs, which are always the final arbiters of inclusion or exclusion in a HyTime BOS.

Some objects are considered by hyperdocument authors to be so critical for proper operation of the hyperdocument that they must be integrated into the effective BOS before the hyperdocument can be considered complete enough for any use to begin. Such critical objects have "foreground priority" in the HyTime BOS. The remaining objects of the HyTime BOS have "background priority". The **bounded object set priority** (*bosprrty*) attribute sets the default priority for itself and for all of the entities in the entity subtree of which it is the root.

NOTE 101 For example, it may be necessary to comply with activity policies appearing in objects that are external to the hub document; such objects should be specified as having foreground priority.

NOTE 102 Applications may choose to assemble the HyTime BOS or not. Similarly, if they do assemble the HyTime BOS, the objects can be assembled in any order. The concept of BOS priority provides a way to suggest a performance optimization for applications that, due to time constraints, must assemble the "foreground priority" objects (objects that are essential to making a minimally adequate presentation of the hyperdocument) first. Once the foreground priority objects of the HyTime BOS have been incorporated into the effective BOS, access to the hyperdocument could begin, while the remaining "background priority" objects are still being incorporated.

The hub document always has, in effect, foreground priority in the HyTime BOS. For all objects in an entity tree, the default priority is specified by the "bosprrty" common notation data attribute of the root entity of that tree. The bosprrty attributes of child entities, if present, specify the default priority for themselves and their children.

NOTE 103 For any given set of objects in an entity tree, the default priority specification may be overridden by one or more bosspec elements referenced by the bosspec attribute of HyDoc (see *6.5 HyTime Bounded object set*).

An entity declared by a hub document whose **subhub** attribute's value is "subhub" is not subject to the normal entity tree discovery process from the perspective of the hub document. Instead, it is regarded as an independent hub document with its own normal entity tree discovery process, whose HyTime BOS is added to the hub document's tentative HyTime BOS. The subhub's maxbos, boslevel, inbos, and bosspec specifications are all taken into account before the resulting subhub HyTime BOS is added to the hub document's HyTime BOS. The hub document specifies that an entity should be considered a subhub by specifying a value of "subhub" for the subhub common notation data attribute. The maxbos and boslevel of the parent hub document are ignored by the subhub, which defines its own, local maxbos and default boslevel.

NOTE 104 A hyperdocument author may wish to include another hyperdocument in the new hyperdocument without having to worry about whether or not his own hyperdocument's boslevel (etc.) will inadvertently render that included hyperdocument's

BOS incomplete, without requiring the author to duplicate and edit that hyperdocument's (perhaps very complex) BOS rules in the new hub document. The subhub facility satisfies this requirement.

NOTE 105 Like all other objects reported by the entity tree discovery process, objects added to the HyTime BOS by virtue of their membership in the HyTime BOS of a subhub document can be removed from the HyTime BOS by bosspec elements referenced by the bosspec attribute of HyDoc.

NOTE 106 Subhubs can declare subhubs.

```
             <!-- HyTime BOS Control Data Attributes -->
<![ %bos; [
<!attlist #NOTATION
-- bosdatt --     -- HyTime BOS control data attributes --
                  -- Clause: 6.5.2 --
   #ALL

   boslevel       -- BOS level --
                  -- Bounded object set level for the entity --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=this entity only) --
      #IMPLIED    -- Default: value of boslevel attribute of HyDoc
                     element. --

   inbos          -- Include in BOS --
                  -- Unconditional include in, or exclude from, BOS --
      (inbos|notinbos)
      #IMPLIED    -- Default: inclusion controlled by BOS level --

   bosprrty       -- Bounded object set priority --
                  -- Default BOS priority of objects in entity tree
                     rooted at this entity. --
      (foregrnd|backgrnd)
      foregrnd

   subhub         -- Is entity a subhub? --
      (subhub|nosubhub)
      nosubhub
>
]]><!-- bos -->
```

6.5.3 Bounded object set exception specification

The **bounded object set exception specification** (*bosspcat*) attribute form specifies bosspec elements that define adjustments to be made to the results of all normal entity tree discovery and subhub processing.

```
<![ %bosspec; [
<!attlist
-- bosspcat --    -- BOS except specification attributes --
                  -- Clause: 6.5.3 --
   (HyDoc)
   bosspec        -- Bounded object set exception specification --
                  -- Adjustments to be made to the bounded object
                     set. --
      IDREFS      -- Reference --
                  -- Reftype: bosspec+ --
                  -- Constraint: must be internal reference --
      #IMPLIED    -- Default: no BOS exception specification --
>
]]><!-- bosspec -->
```

The element form **bounded object set exception specification** (*bosspec*) identifies an entity in the BOS rooted at the hub document that refers to the bosspec in order to control its inclusion in the BOS, its “priority”, or both. Bosspec elements are resources used by reference via the bosspec attribute of the HyDoc element form. A bosspec element must occur in the hub (or subhub) document that refers to it.

A bosspec element contains one or more **BOS paths**. A BOS path consists of a list of entity names, of which the first must be declared in the hub (or subhub) document that contains the bosspec. The next entity name is the name of an entity declared within that entity, etc. The last entity named in a BOS path is the entity being addressed. When there is more than one path in a bosspec, each path is enclosed in parentheses. Entity names that do not conform to the concrete syntax of the hub document may be specified as literals if necessary.

The boslevel, inbos, and bosprrty attributes are as defined for the common data attributes of the same names (see *6.5.2 HyTime BOS control data attributes*).

The hub document's bosspecs are applied after all subhub and non-subhub entity tree discovery has been completed. The final adjustments to the HyTime BOS are made by the hub document's bosspecs, and of those adjustments, the ones whose inbos attribute's value is “notinbos” are applied last of all, so they can exclude any objects from the HyTime BOS, regardless of how they came to be tentatively included in it.

NOTE 107 Conceptually, every subhub's BOS is established *before* the BOS of the hub or subhub with respect to which it is a subhub. Therefore, conceptually, the sub-most subhub's BOS is established first, and the hub's BOS established last, so that the hub's bosspecs are the final arbiters. Whatever is contributed to the hub's BOS (or any subhub's BOS) by the hub's (or subhub's) own subhubs can be discarded selectively by the hub's (or subhub's) own bosspecs. Each hub or subhub applies its own bosspecs as part of the establishment of its own BOS, and these can be used to exceptionally include or exclude any entity included in or excluded from the BOS established by combining the BOSs of all of its own subhubs, together with the BOS established by its own entity declarations, boslevel, maxbos, etc. The bos control attributes do not apply to the contributions to the BOS that are the BOSs of subhubs. A subhub's BOS is incorporated wholesale, and only a bosspec can override it.

If the same storage object is declared as an entity more than once in the BOS, excluding one of its declarations with a bosspec will not automatically exclude its other declarations; complete exclusion of the storage object requires exclusion of all the entities for which it is the replacement data.

NOTE 108 In other words, there is no SGML- or HyTime-defined mechanism for determining that two different entities are in fact the same storage object (although processing applications are free to define their own algorithms for doing so).

```
         <!-- Bounded Object Set Exception Specification -->
<![ %bosspec; [
<!element
   bosspec        -- Bounded object set exception specification --
                  -- Clause: 6.5 --
                  -- Used to affect the HyTime BOS by overriding the
                     inclusion or exclusion and priority of the
                     entities identified by the BOS path or paths
                     given as content. --
   - O
   (#PCDATA)      -- Lextype: ((ENTITY,(csname|literal)*)|
                              (GRPO,ENTITY,(csname|literal)*,GRPC)+)
                     --
                  -- Constraint: If parentheses are used, each
                     parenthesized list is a separate BOS path. --
                  -- Constraint: Each word or literal in a BOS path is
```

```
                       the name of an entity declared in the entity
                       identified by the previous word, literal, or
                       entity name. --

-- Attributes [base]: bosspec --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: HyDoc:bosspec --
>
<!attlist
   bosspec        -- Bounded object set exception specification --
                  -- Clause: 6.5 --

   boslevel       -- BOS level --
                  -- The BOS level from the last entity named in
                     each specified BOS path to be affected by this
                     bosspec. --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=last entity only) --
      1

   inbos          -- Include in BOS --
                  -- Unconditionally include or exclude objects
                     declared by the last entity named in each BOS
                     path, to the BOS level specified by this
                     bosspec's boslevel attribute. --
      (inbos|notinbos)
      #IMPLIED    -- Default: BOS unaffected --

   bosprrty       -- Bounded object set priority --
                  -- Unconditionally specify the BOS priority of
                     objects declared by the last entity named in each
                     BOS path, to the BOS level specified by this
                     bosspec's boslevel attribute. --
                  -- Note: The semantic of the bosprrty attribute is
                     not affected by the value of the inbos attribute
                     (that is, whether it is explicitly "inbos" or the
                     value is implied). --
      (foregrnd|backgrnd)
      #IMPLIED    -- Default: no priority change --
>
]]><!-- bosspec -->
```

6.6 HyTime architectural bridging forms

The element form **HyTime architectural bridging element** (*HyBrid*) bridges between HyTime and non-HyTime elements. No HyTime processing is defined for the HyBrid form other than that provided by the HyTime common attributes. The HyBrid form must be used for any element type that uses any of the HyTime common attributes but does not have a more specific HyTime element form. When architectural markup minimization is in effect, the HyBrid form is automatically used for any client element that is considered architectural (because it has a unique identifier) but that does not have a more specific HyTime form (see *A.3.6.2 Architectural markup minimization*).

```
            <!-- HyTime Architectural Bridging Element -->
<!element
   HyBrid         -- HyTime architectural bridging element --
                  -- Clause: 6.6 --

   - O
   (%HyCFC;)*

-- Attributes [base]: HyBrid --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   HyBrid         -- HyTime architectural bridging element --
                  -- Clause: 6.6 --

   GenArc   NAME     #FIXED GABrid
>
```

The notation form **HyTime architectural bridging notation** (*HyBridN*) bridges between HyTime and non-HyTime data entities. No HyTime processing is defined for the HyBrid form other than that provided by the HyTime common data attributes. The HyBridN form must be used for any data entity that uses any of the HyTime common data attributes but does not have a more specific HyTime notation form. When architectural markup minimization is in effect, the HyBridN form is automatically used for any client data entity that does not have a more specific HyTime form (see *A.3.6.2 Architectural markup minimization*).

```
           <!-- HyTime Architectural Bridging Notation -->
<!notation
   HyBridN        -- HyTime architectural bridging notation --
                  -- Clause: 6.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Architectural Bridging Notation//EN"

-- Attributes [base]: HyBridN --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyBridN        -- HyTime architectural bridging notation --
                  -- Clause: 6.6 --

   GenArc   NAME     #FIXED GABridN
>
```

6.7 Common attributes

A set of architecture control attributes for all architectures is available to HyTime applications (see *A.3.5 Architecture control attributes*), as are the common attributes of the General Architecture (see *A.5 General*

Architecture), and, when the location address module is supported, the ID reference control attributes (*7.7 Reference control*). This clause defines additional attributes for the HyTime architecture.

NOTE 109 By default, the HyTime meta-DTD includes the declarations necessary to use the “id” attribute of the General Architecture. Use of other General Architecture facilities must be declared in the documents or architectural meta-DTDs that use them.

NOTE 110 Elements that use the common attributes must also conform to a defined HyTime architectural form. The HyBrid form can be used for those elements that would otherwise not be clients of a more specific HyTime element form.

6.7.1 Value Reference

The attribute form **value reference** (*valueref*) associates attributes, an element's content, or an element itself, with referential attributes (or content) that can be used to refer to the effective value of the attribute, content, or element node. The referential attributes so named are referred to collectively as “value reference attributes”.

NOTE 111 The “effective value” of an attribute is the value that an application or architectural processor needs, which is not necessarily limited to the types of data or objects that can be expressed through SGML's attribute value syntax. For example, an element type may represent an application-defined object class that has properties that are themselves structured. The “property-of” relationship can be indicated by making the property an attribute of the element type. The structure of the property can be represented by using additional elements to structure the value and using value reference to refer to those elements as the “real” value of the attribute (where the syntactic value of the attribute might be the reference itself).

NOTE 112 A “#ELEMENT” value reference allows one SGML element to act as a “placeholder” for another element held somewhere else (for example, as a subdocument). However, the element making the reference is still addressable. Addressing the element making the reference is equivalent, for the purposes of HyTime-specific processing, to addressing the effective element node.

The valueref attribute value is a list of name pairs. The first name of each pair may be an attribute name, the keyword “#CONTENT”, or the keyword “#ELEMENT”. It names the attribute, content (“#CONTENT”), or complete element (“#ELEMENT”) for which the value is to be addressed by the attribute or content named by the second keyword. If the same attribute (or #CONTENT) is used for both names, the attribute or content refers to its own semantic value and never takes its semantic value directly (e.g., the exspec attribute of the event element form). The keywords “#CONTENT” and “#ELEMENT” may only appear once as the first keyword of a pair. The keyword “#CONTENT” may only appear once as the second keyword of a pair.

When the first name in a pair is “#CONTENT”, the effective value of the content property of the element is the node list addressed by the attribute or content named by the second name in the pair.

NOTE 113 For example, to provide an attribute called “conloc” for referring to the content of an element, use a declaration of the form:

```
<!ATTLIST Para
   HyTime   NAME     #FIXED "HyBrid"
   valueref CDATA    #FIXED "#CONTENT conloc"

   -- Locate the content of the para --
   conloc   IDREF    #IMPLIED
>
```

The valueref attribute indicates that the content of the element Para will be located by reference using the referential attribute “conloc” (the conloc attribute is referential because it has a declared value prescription of “IDREF”).

When the first name in a pair is “#ELEMENT”, the effective element node is the single node addressed by the attribute or content named by the second name in the pair. When the object addressed is a document or subdocument entity, the effective element node is the document element of the addressed document. It is an RHE to address more than one node as the effective element node.

The objects located as the effective value of attributes, content, or elements need not satisfy the syntactic constraints defined in the referring document's document type declaration. However, they must satisfy any semantic constraints defined by the processing application. The mechanism for defining such constraints is outside the scope of the HyTime standard.

NOTE 114 Semantic constraints could include conformance to the lexical type of an attribute, the content model of the referring element, conformance to one or more architectural meta-content models, or other semantic constraints unrelated to SGML-defined constraints. Application designers can use the reference type control facilities of HyTime and the General Architecture to control the types and organization of elements referred to.

Value reference does not modify how an SGML document is parsed.

NOTE 115 Conceptually, the result of a value reference is maintained in the HyTime semantic grove, not the HyTime effective SGML document grove. This means in particular that a node addressed by value reference does not occur in any name spaces in the HyTime extended SGML document grove of the document from which the value reference is made if the node's origin is not already in that grove.

Thus, if an element in another document is made the content of an element by value reference, the referenced element can be addressed by name only by a inter-document address but can be addressed directly by tree location with respect to the referring element.

For the purposes of accessing attribute values and the content of elements via HyTime location addressing, the data addressed by value reference are always used even if there is a directly-specified value for the attribute or content. The only exception is property location addresses, which can, optionally, address the direct value of the attribute or element, rather than its referenced value.

NOTE 116 In other words, valueref serves to “redirect” location addresses from the direct value to the indirectly-addressed value.

Addresses resolved by means other than a HyTime engine (such as a DSSSL engine) need not respect the redirection performed by value reference attributes.

When the location addressing module is supported, any supported means of addressing may be used for value reference. If location addressing is not supported, value reference attributes must be declared as IDREF, IDREFS, ENTITY, or ENTITIES and content cannot be used for value reference.

An element that uses value reference to refer to its content may allow syntactic content. There is no requirement that elements that provide an attribute for value reference of their effective content declare that attribute with a default value of #CONREF — any syntactic content will be ignored for HyTime processing when the value reference attribute is specified in any case (unless the content itself contains the reference).

NOTE 117 In the conloc example above, the Para element could have syntactic content, but HyTime location addresses that address the content of the Para element would address the objects addressed by the Para's conloc attribute, not the data in its syntactic content (except as provided by the proploc element form).

```
             <!-- Value Reference Attribute Namer -->
<![ %valueref; [
<!attlist
-- valueref --    -- Value reference attribute namer --
                  -- Clause: 6.7.1 --
   #ALL

   valueref       -- Value reference attribute namer --
                  -- Associates attributes (or element's content) with
                     referential attributes (or content) that can be
                     used to refer to the semantic value of the
                     attribute or content. --
      CDATA       -- Lextype: ((ATTORCON|"#ELEMENT"),ATTORCON)+ --
                  -- Note: The first ATTORCON is the attribute or
                     content for which the value is to be addressed,
                     the second ATTORCON is the attribute or content
                     by which the value reference is made. --
                  -- Constraint: second attribute in each pair must be
                     a referential attribute or content defined as
                     referential by the refloc facility. --
                  -- Constraint: "#ELEMENT" and "#CONTENT" may only
                     occur once as first keyword of pair.  "#CONTENT"
                     may only occur once as second keyword of pair. --
      #IMPLIED    -- Constant --
>
]]><!-- valueref -->
```

The following form is included for backward compatibility.

```
                      <!-- Content Location -->
<![ %conloc; [
<!attlist
-- conloc --      -- Content Location --
                  -- Clause: 6.7.1 --
   #ALL

   HyBase   NAME     #FIXED HyBrid
   valueref CDATA    #FIXED "#CONTENT conloc"

   conloc         -- Content location --
                  -- Location of the element's semantic content. --
      CDATA       -- Reference --
      #CONREF     -- Default: syntactic content --
>
]]><!-- conloc -->
```

6.7.2 Descriptive text

The descriptive text facility is an alternative form of value reference that allows elements that are frequently repeated in a document to be accessed by a descriptive name in a lookup table. The lookup table can also be thought of as providing a definition for the descriptive text, where the definition is the content and attributes of the element described.

6.7.2.1 Descriptive text attributes

The attribute form **descriptive text attributes** (*dtxtatt*) allows the content and attributes of an element to be specified by a descriptive name. All of its attributes must be defined if the form is used.

The attribute **descriptive text** (*desctxt*) is a character string that describes a frequently-used element. The descriptive text is normalized to remove extraneous white space and facilitate table lookup.

If the attribute is specified, the content of the element must be empty. The description tables on the current list are searched for the first occurrence of the descriptive text. If one is found and its definition is of the same element type as this element, the definition's content and attributes (other than its unique identifier and descriptive text attributes) are used with the unique identifier (if any) of this element. Alternatively, if the definition satisfies the content model of this element, the definition is treated as the content. It is an RHE if the descriptive text is not found in the current tables, or if the definition of the text that is found does not satisfy either of the above conditions.

The attribute **description table** (*desctab*) specifies an ordered list of description tables that, in order, will be used to look up descriptive text. The list will remain in use until changed by another occurrence of this attribute on the same element type (or another of the set of element types that share the attribute list, if any). The order of occurrence is the order in the SGML representation of the document.

NOTE 118 Note the distinction between descriptive text and the value reference facility. Both allow the content of the element to be outside the syntactic content. However, valueref locates the content in another object that may exist for its own purpose, while desctxt draws it from a table of elements that were created specifically as a resource for use by reference.

```
                 <!-- Descriptive Text Attributes -->
<![ %desctxt; [
<!attlist
-- dtxtatt --      -- Descriptive text attributes --
                   -- Clause: 6.7.2.1 --
   #ALL

   desctxt         -- Descriptive text --
                   -- If specified, its descdef in desctab is treated
                      by HyTime as the element or as its content. --
      CDATA        -- Lextype: words --
      #CONREF      -- Default: syntactic content (and attributes) --

   desctab         -- Description table --
                   -- Current description tables --
      CDATA        -- Reference --
                   -- Reftype: desctab+ --
                   -- Constraint: searched in order listed --
      #IMPLIED     -- Default: use previously specified set of
                      tables --
>
]]><!-- desctxt -->
```

6.7.2.2 Description table

The element form **description table** (*desctab*) is a lookup table that associates descriptive text strings with definitions of attributes and content.

NOTE 119 For example, the string “long pause” could be defined by an event element that has an extent of 3 seconds and contains a “pause” element.

The first occurrence of a descriptive text in the table is used. Subsequent occurrences are ignored, but are not erroneous.

NOTE 120 A style sheet could contain description tables that redefine the descriptive text.

```
                      <!-- Description Table -->
<![ %desctxt; [
<!element
   desctab         -- Description table --
                   -- Clause: 6.7.2.2 --
                   -- Descriptive text definition table --
   - O
   (desctxt,descdef)+

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: dtxtatt:desctab --
>
]]><!-- desctxt -->
```

6.7.2.3 Descriptive text

The element form **descriptive text** (*desctxt*) is a character string that is a valid value for the attribute “descriptive text” (see *6.7.2.1 Descriptive text attributes*).

```
                      <!-- Descriptive Text -->
<![ %desctxt; [
<!element
   desctxt        -- Descriptive text --
                  -- Clause: 6.7.2.3 --
   O O
   (#PCDATA)      -- Lextype: words --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
]]><!-- desctxt -->
```

6.7.2.4 Descriptive text definition

The element form **descriptive text definition** (*descdef*) contains one or more elements that define a descriptive text by providing the necessary attributes and data. The allowable elements are determined by the context in which the descriptive text is used.

```
                 <!-- Descriptive Text Definition -->
<![ %desctxt; [
<!element
   descdef        -- Descriptive text definition --
                  -- Clause: 6.7.2.4 --
   O O
   (%HyCFC;)*

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
]]><!-- desctxt -->
```

6.7.3 Activity policy association

In interactive systems the data objects available for interaction have access policies associated with them, either implicitly or explicitly. In many systems, access policies are implicit in the design of the system or expressed through system facilities such as user-based access permissions. Policies may also be associated with objects explicitly by associating activity policy definitions with the objects to which the policies apply. Typically, the owners of information objects associate owner policies with them. For example, a policy to protect the ownership of a piece of intellectual property may require the purchase of a license before access is permitted.

The HyTime activity policy association facility is a standardized means of expressing the attachment (“association”) of activity policies to information objects. Such policies take effect and are applied (“triggered”) whenever users, authors and editors attempt to engage in defined classes of interaction with information objects. “Activity” is the generic term for all such interaction; each class of interaction is an “activity type”. HyTime standardizes neither the expression nor the management of the details of activity policy specification and enforcement. The definition of activity policies is application specific and may consist of any combination of modes and manners of definition, including legal agreements expressed in natural language and interactive computer programs.

NOTE 121 A HyTime engine that supports the activity option may provide the service of managing the associations between the policies and the objects governed by such policies, but it may have no ability to understand or enforce the policies themselves. A complete HyTime system may make use of such an infrastructural policy association management service by distinguishing between classes of activity (e.g., access to the governed objects, versus modification of the governed objects), asking the HyTime engine for the policies that are applicable to the particular class of activity, and bringing those policies to bear in some way, perhaps even enforcing them. Alternatively, a HyTime engine may understand and even enforce certain kinds of policies, and the implementers of a complete HyTime system that incorporates such a HyTime engine may therefore not be required to develop activity policy triggering and enforcement features.

Activity policy association rules are associated with information objects in two ways: objects that are SGML elements can refer to the activity policy rules that govern them using the actrules attribute form, and activity policy rule (actrule) elements can refer to any information objects, including but not limited to SGML elements. In either case, the activity policy is automatically associated not only with the information object itself, but also with its metadata (that is, in the case of an SGML element, its attribute values) and with all of the components of its content.

Activity policies can be associated with any objects in any notations; their applicability is not limited to objects in HyTime or SGML notation, and the HyTime-defined activity types are applicable even when the activity is accomplished by means not provided by HyTime or SGML (see note note 126).

The common attribute form **activity policy association rules** (*activity*) refers to one or more activity policy association rules by which the element making the reference is governed.

```
              <!-- Activity Policy Common Attributes -->
<![ %activity; [
<!attlist
-- activity --    -- Activity policy association rules --
                  -- Clause: 6.7.3 --
   #ALL

   actrules       -- Activity policy association rules --
                  -- Activity policy association rules that apply to
                     this element. --
      CDATA       -- Reference --
                  -- Reftype: actrule* --
      #IMPLIED    -- Default: No activity policy association rule is
                     specified by this element as governing this
                     element. --
>
]]><!-- activity -->
```

The element form **activity policy association rule** (*actrule*) defines association rules. Actrule elements can be used to declare associations between particular information objects and particular policies, or to declare that particular information objects are dissociated from particular policies, thus possibly nullifying some portions of the effectivity of other actrule elements. The act of associating or dissociating policies is referred to generically as “applying” policies to objects, even though the ultimate effect may be for policies to be dissociated from objects. In addition, an actrule can be used to nullify all of the associations or dissociations declared by other actrules.

The policies associated are distinguished by the type of activity performed on the governed object. Any types of activity may be distinguished from any other types in any application-defined way. HyTime defines a set of activity types; these or others may be used in any combination.

NOTE 122 The primary purpose of distinguishing activity types is to allow a performance optimization in which only the policies that are applicable to a particular kind of activity need be triggered, retrieved, and enforced by an application. Without such distinctions, all policies would always necessarily be triggered by every attempt to engage in any sort of activity. In general, if there are particular classes and/or subclasses of activity that are commonplace in a given application and which are governed by distinct sets of policies, the declaration and use of an application-specific set of activity types, with or without some or all of the HyTime-defined default set, should be considered.

The HyTime-defined activity policy types are:

create **Object creation policies** that were applied when the object was created.

NOTE 123 Object creation activity is different from all other classes of activity because it is not triggered by a user's attempt to do anything. Rather, it exists because it is often useful to know what policies did in fact govern the creation of an object; such policies might indicate where the object was cross-listed, how and where information about the creation of the object was stored, etc.

access **Object access policies** that are applied whenever an attempt is made to read, inspect, or otherwise use or access an object or any node in the subnode tree of which the object is the root.

NOTE 124 When an object is copied, the access policies must necessarily also be applied in addition to the copying policies, because it is impossible to copy something without accessing it. (It is possible, although perhaps comparatively unusual, for modification or deletion of objects to occur without first accessing them, but it seems likely that access policies should also often be triggered in cases where modification or deletion policies are triggered.)

copy **Object copying policies** that apply when attempts are made to copy the objects.

delete **Object deletion policies** that apply when attempts are made remove the objects from the document(s) in which they exist.

NOTE 125 The deletion of an object from a document is tantamount to its destruction, at least from the perspective of the document from which it is deleted. Object deletion policies should be crafted in such a way as to account for the possibility of actual destruction of the deleted information.

modify **Object modification policies** that are applied whenever an attempt is made to modify an object, including the modification or deletion of any of the nodes of the subnode tree of which the object is the root.

link **Object hyperlinking policies** that are applied whenever an attempt is made to specify an object (or any of the nodes of the subnode tree of which the object is the root) as the anchor of any hyperlink. For example, object hyperlinking policies may be triggered either by an attempt to create a hyperlink to the object, or by an attempt to add a HyTime document containing hyperlinks to the object, to a BOS.

unlink **Object unhyperlinking policies** that are applied whenever an attempt is made to delete an existing specification of an object (or any of the nodes of the subnode tree of which the object is the root) as an anchor of a hyperlink. For example, an object unhyperlinking policy may be triggered either by an attempt to delete a hyperlink to an object, or by an attempt to remove a document containing hyperlinks to an object from a BOS.

schdul **Object scheduling policies** that are applied whenever an attempt is made to schedule an object, modifier, or projectr (or any of the nodes of the subnode tree of which it is the root) in an FCS. For example, an object scheduling policy may be triggered either by an attempt to schedule an object, or by an attempt to add a HyTime document, containing scheduled events that schedule the object, to a BOS.

unschdul **Object unscheduling policies** that are applied whenever an attempt is made to remove a scheduled object, modifier, or projectr (or any of the nodes of the subnode tree of which the object is the root) from a logical FCS. For example, an object unscheduling policy may be triggered either by an attempt to remove an object from an FCS, or by an attempt to remove from a BOS a HyTime document containing scheduled events that schedule the object.

reschdul **Object rescheduling policies** that are applied whenever an attempt is made to change the effective extent of the appearance of an object (or any of the nodes of the subnode tree of which the object is the root) in an FCS, without removing or adding any appearance of the object in an FCS. (Otherwise, schdul or unschdul policies may apply instead.)

trnscl **Object transclusion policies** that are applied whenever an attempt is made to incorporate an object (or any of the nodes of the subnode tree of which the object is the root) by reference into any other document or hyperdocument (e.g., by the use of entity reference, valueref attributes, or hyperlinks whose application semantic is transclusion).

untrnscl **Object untransclusion policies** that are applied whenever an attempt is made to make an object (or any of the nodes of the subnode tree of which the object is the root) no longer incorporated by reference into any other document or hyperdocument.

NOTE 126 Object hyperlinking and unhyperlinking policies apply even when the hyperlink being created is not expressed in HyTime or SGML notation.

Object scheduling, unscheduling, and rescheduling policies apply even when the scheduling facilities used do not conform to the semantics of HyTime fcs (and related) elements, and even when the scheduling information is not expressed in HyTime or SGML notation.

HyTime and SGML provide several mechanisms whereby transclusion may occur, including SGML entity references, HyTime valueref attributes, and HyTime hyperlinks whose application-defined semantic is one of transclusion. If the effect of an activity is in any sense to make part or all of one document appear in any way as though it were part of another, regardless of the mechanism used, regardless of whether the mechanism used is an SGML or HyTime mechanism, object transclusion and untransclusion policies apply.

NOTE 127 Activity policies may take into account the BOS status of the objects that play a role in various types of activity. For example,

— An object hyperlinking policy may ignore hyperlinking activity if it consists only of adding to the effective BOS a preexisting document which contains both the object and the hyperlink, or if the object and the hyperlink are both included in the HyTime BOS of the hyperdocument derived according to the specifications of the hub document.

— An object unhyperlinking policy may ignore unhyperlinking activity if it consists only of the removal from the effective BOS of a document that is not a member of the HyTime BOS of the hyperdocument derived according to the specifications of the hub document.

HyTime defines each of the above activity types, and, by default, each of them has a corresponding “activity type” attribute of the same name, which is also mentioned in the default value of the actypes attribute. Each of the activity type attributes specifies the policy definitions that are to be applied whenever the activity of that type involving the objects governed by the rule is attempted.

Activity policy associations can convey several many-to-many relationships. A single activity policy can comprise many sets of conditions under which a given activity will or will not be permitted, and many sets of actions that will be taken if an attempt is made to perform any activity governed by such an activity policy. Many activity policies may govern one or many activities, and a single activity policy can govern any combination of activity types.

The objects specified by the governed attribute of an actrule must be bound to their governing activity policies before any activity is permitted throughout the effective BOS. The objects specified by activity types attributes of actrules and by the actrules attribute of other elements can be bound at any time prior to the first activity involving any objects whose governing activity policies are associated by those attributes. If the activity option of the Base

Module is supported, it is an RHE if any governed, actrules, or activity type attribute of an actrule element cannot be fully resolved when any attempt is made to resolve it.

The attribute **governed objects** (*governed*) refers to the objects to which the policies specified by the activity type attributes are to be applied.

The attribute **associate or dissociate** (*associat*) indicates whether, for the governed object(s), this actrule associates (assoc) or dissociates (dissoc) the policies specified by the activity type attributes.

The attribute **nullify activity policy association rules** (*nullify*) nullifies the effectivity of the associations (or dissociations) established in the actrule(s) to which it refers. However, the actrule nullifications specified by a nullified actrule remain in effect.

The combination of all associated (but not expressly dissociated) policies expressed by all unnullified actrule elements apply to any given information object, regardless of whether the associations were made with the governed attribute of one or more actrule elements, or by the actrules attributes of the governed object or of one or more ancestor elements, or by both means of specifying association. The order in which the policies are applied is not defined by this international standard. All applicable policies must be satisfied before an attempt to perform an activity is permitted to succeed. Policies that are associated more than once are effective in precisely the same way as policies that are associated only once.

NOTE 128 If the same policy is redundantly specified, depending on how the policy is redundantly referenced, a HyTime engine may or may not be able to eliminate the redundancy before reporting the policies governing an information object. It may be incumbent on other parts of the system to recognize redundancy and eliminate it.

As a practical matter, in order for an activity policy to take effect, the governed information object, the activity policy, and the actrule that associates the policy with the object must all be present in the effective BOS, and no dissociating or nullifying actrule can be in the effective BOS. Therefore, whoever (and/or whatever) controls the membership in the effective BOS also, in effect, has control over the application of activity policies.

NOTE 129 This means, for example, that if Party B purchases all rights to a hyperdocument from Party A, and the hyperdocument as delivered to B contains some set of governing activity policies, B does not have to edit the hyperdocument in order to change the effectivity of the preexisting activity policies and/or replace them with new ones. Instead, all that B needs to do is to add a new hub document to the hyperdocument that declares the old hub as a subhub, and which contains actrules that nullify (or partially nullify) A's old actrules, and which provides for the inclusion in the effective BOS of B's new actrules and policies (perhaps by including them in the HyTime BOS).

Of course, Party B can only do this if the original hyperdocument does not have activity policies that prevent association of new activity policy rules. In addition, this example assumes that both parties are operating within a system that enforces the activity policies of both Parties, such as within an enterprise's intranet.

The ability to override existing activity rules does not mean necessarily that the rules defined in one hub document can be overridden by making that hub document a subhub of another hub document. This is because the only way in which activity policies can be enforced at all is if access to the members of a hyperdocument is through logical references made from the hub document to the other parts of the hyperdocument, either directly or indirectly (e.g., by hyperlinking from the hub document to the other documents in the hyperdocument). These references will necessarily trigger the corresponding activity policies of the subhub hyperdocument, if any.

NOTE 130 HyTime does not necessarily require that all access to members of a hyperdocument be made through references made by the hub document — it is merely the practical reality that access must begin with the hub document if activity policies are to be effectively enforced. In other words, activity policies can only be enforced within a closed system with the necessary access controls. However, activity policies are still of value even in uncontrolled environments because they communicate the desired policy, whether or not it can be enforced by the access system (for example, a shareware license policy associated with a program available for anonymous FTP).

The **activity types** (*actypes*) attribute's value is the list of the names of the activity type attributes for the client element type. The default value is the complete list of the activity types defined by HyTime. In any given client element type, the list can include all, some, or none of the HyTime-defined activity types, along with any number of application-defined activity types, each of which is also declared as an attribute of the client element type. If any of the HyTime-defined activity types are used, their semantics are as defined in this international standard. The actypes support option of the base module must be supported if any activity types other than those defined in this International Standard are declared.

```
                  <!-- User activity types -->
<![ %actypes; [
   <!entity %
      dactypes    -- Are default activity types fixed? --
                  -- Clause: 6.7.3 --

      ""          -- Activity types not fixed --
   >
   <!ENTITY % activity "INCLUDE" >
]]><!-- actypes -->

              <!-- Activity policy association rule -->
<![ %activity; [
<!entity %
   dactypes       -- Are default activity types fixed? --
                  -- Clause: 6.7.3 --

   "#FIXED"       -- Activity types are fixed --
>
<!element
   actrule        -- Activity policy association rule --
                  -- Clause: 6.7.3 --
                  -- Optionally establishes association or
                     dissociation of activity policies with
                     information objects, and optionally nullifies
                     the effectivity of other actrules. --
   - O
   (%HyCFC;)*

-- Attributes [base]: actrule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: activity:actrules, actrule:nullify --
>
<!attlist
   actrule        -- Activity policy association rule --
                  -- Clause: 6.7.3 --

   governed       -- Governed objects --
                  -- Object(s) with which are associated or
                     dissociated the policies specified by the set of
                     activity type attributes of this actrule. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No objects are specified by this rule
                     as being governed by this rule. --

   associat       -- Associate or dissociate --
                  -- Specifying "assoc" creates associations, "dissoc"
                     dissolves associations created by other
                     actrules. --
      (assoc|dissoc)
      assoc

   nullify        -- Nullify activity policy association rules --
                  -- Nullify the effectivity of the actrules
                     referenced. --
      CDATA       -- Reference --
                  -- Reftype: actrule* --
      #IMPLIED

   actypes        -- Activity types --
                  -- List of the names of activity type attributes. --
                  -- Note: Any, all, or none of the members of the
                     HyTime default activity type set may be used, in
                     addition to any application-defined set. --
                  -- Constraint: Cannot redefine semantics of members
                     of the HyTime-defined default set. --
      NAMES       -- Lextype: ATTNAME+ --
      %dactypes;  -- Note: Will be #FIXED if actypes not supported --
      "access copy create delete link modify reschdul schdul trnscl
       unlink unschdul untrnscl"
                  -- Constant --

-- The following are the twelve HyTime-defined "activity type"
   attributes.  Use of these names always invokes their
   HyTime-defined semantics. --

   create         -- Object creation policies --
                  -- Policies that applied when any of the objects
                     were originally created. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object creation policies specified --

   copy           -- Object copying policies --
                  -- Policies that apply when attempts are made to
                     copy any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object copying policies specified --

   delete         -- Object deletion policies --
                  -- Policies that apply when attempts are made to
                     delete any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object deletion policies specified --

   access         -- Object access policies --
                  -- Policies that apply when attempts are made to
                     access any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object access policies specified --

   modify         -- Object modification policies --
                  -- Policies that apply when attempts are made to
                     modify any of the objects --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object modification policies
```

```
                     specified --

   link           -- Object hyperlinking policies --
                  -- Policies that apply when attempts are made to
                     make any of the objects the anchor of a
                     hyperlink. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object hyperlinking policies
                     specified --

   unlink         -- Object unhyperlinking policies --
                  -- Policies that apply when attempts are made to
                     make any of the objects no longer the anchor of a
                     hyperlink. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object unhyperlinking policies
                     specified --

   schdul         -- Object scheduling policies --
                  -- Policies that apply when attempts are made to
                     schedule any of the objects in an FCS. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object scheduling policies
                     specified --

   unschdul       -- Object unscheduling policies --
                  -- Policies that apply when attempts are made to
                     remove any given appearance of any of the objects
                     in an FCS. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object unscheduling policies
                     specified --

   reschdul       -- Object rescheduling policies --
                  -- Policies that apply when attempts are made to
                     change the effective extent of any appearance of
                     any of the objects in an FCS. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object rescheduling policies
                     specified --

   trnscl         -- Object transclusion policies --
                  -- Policies that apply when attempts are made to
                     transclude any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object transclusion policies
                     specified --

   untrnscl       -- Object untransclusion policies --
                  -- Policies that apply when attempts are made to
                     make any of the objects no longer transcluded. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object untransclusion policies
                     specified --
>
]]><!-- activity -->
```

6.8 Coordinate Specifications

A “coordinate address” is a means of specifying a location by the relative position of an object. A coordinate address is only meaningful in the context of a “coordinate space”; that is, a set of coordinate axes and a system for measuring along them.

The location addressing and scheduling modules of HyTime both use coordinate specifications to address objects and specify the positions of objects within coordinate spaces. Support for coordinate specifications is an option of the base module that is required (and implied) by some location address element forms and by the scheduling module in general.

Conceptually, the phrase “the third and fourth words in the sentence” is an example of locating an object in a coordinate space. The coordinate space has one axis, that of the sentence. For measurement, the axis is divided into indivisible “quanta” which can be counted. In this case, the quantum is a word.

An axis, then, is an ordered set of quanta. An object is located on an axis by its “dimension”, which consists of two components: a position along the axis and a quantum count (the total number of quanta occupied by the object). The position of an object along an axis is given as the quantum number of either the first or last quantum occupied by the object. The quantum count of an object is either given directly or derived from the quantum numbers of the first and last quantum. In the example, the dimension of “the third and fourth words” can be described by the quantum number of the first quantum (3) followed by the quantum count (2): “3 2”. The quantum number of the last quantum would be 4.

NOTE 131 HyTime uses the term “dimension” as it relates to the length of measurements, as on a mechanical drawing, rather than “dimension” in the sense of “multidimensional space.”

6.8.1 HyTime axis marker list notation

The **HyTime axis marker list** notation defined in this clause is the default way to specify dimensions in HyTime documents.

```
              <!-- HyTime Axis Marker List Notation -->
<![ %HyMrkLst; [
<!notation
   HyMrkLst       -- HyTime Axis Marker List Notation --
                  -- Clause: 6.8.1 --
                  -- A list of HyTime axis markers (signed non-zero
                     integers) and marker functions (elements that
                     conform to the markfun architectural form).
                     Marker functions are evaluated to lists of
                     markers. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Axis Marker List Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
]]><!-- HyMrkLst -->
```

6.8.1.1 Axis marker list

An **axis marker** is an integer other than zero that represents a position on a coordinate axis. A positive value counts from the start of the range, beginning at 1 for the first quantum. A negative value counts from the direction of the end of the range, beginning at -1 for the last quantum.

An **axis marker list** is a list of intermixed axis markers and/or elements that resolve to axis markers.

NOTE 132 When the marker function (markfun) option is supported, axis markers may be specified in documents using a different notation (see *6.8.1.2 Marker Functions*).

The architectural parameter entity *marklist* lists the constructs that can be used anywhere axis markers are allowed. This parameter entity is then used in the definition of other architectural content models.

```
<!entity %
   marklist       -- Axis marker list content model --
                  -- Clause: 6.8.1.1 --
                  -- Data and elements that resolve to lists of
                     markers. --

   "#PCDATA|dimref|markfun"
>
```

6.8.1.2 Marker Functions

A marker function is an element that resolves to a list of zero or more axis markers. The notation used to specify the markers is arbitrary and user-defined, but the processor for the notation must return the list of axis markers to the marklist notation processor.

NOTE 133 In other words, a marker list notation processor that supports marker functions must provide an interface by which marker function notation processors will return axis marker lists to it.

NOTE 134 The **HyTime axis marker list notation**, the **HyTime marker function language** (see *6.8.6 HyTime Marker Function Language (HyFunk)*), and the **HyTime dimension reference notation** (see *9.8 Dimension referencing*) are all marker function notations.

The element form **marker function** (*markfun*) contains an element or data that is interpreted as a function that returns one or more axis markers.

```
                       <!-- Marker function -->
<![ %markfun; [
<![ %HyFunk; [
   <!entity % dmarkfun "HyFunk">
]]>
<!entity %
   dmarkfun       -- Default marker function notation --
                  -- Clause: 6.8.1.2 --

   "#REQUIRED"
>
<!element
   markfun        -- Marker function --
                  -- Clause: 6.8.1.2 --
                  -- Evaluates to a list of zero or more axis
                     markers. --
   O O
   (%HyCFC;)*     -- Constraint: content must conform to specified
```

```
                     marker function notation. --

-- Attributes [base]: markfun --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   markfun         -- Marker function --
                   -- Clause: 6.8.1.2 --

   notation        -- Marker function notation --
      NAME         -- Lextype: NOTATION --
      %dmarkfun;
>
]]><!-- markfun -->
```

6.8.2 HyTime dimension specification notation

A **dimension** is represented as a list of two axis markers. The two members of the pair of markers in the list are known respectively as **marker1** and **marker2**, or the **first marker** and **second marker**.

The sign of each axis marker indicates the direction of counting, as follows:

— First marker

positive   Count forward from start of range (1).

negative   Count back from end of range (-1) if marker2 is positive, else count back from marker2.

— Second marker

positive   Count forward from marker1.

negative   Count back from end of range (-1).

The second marker must represent a position from the first marker through the last quantum of the axis, inclusive.

```
START OF RANGE <= first position <= second position <= END OF RANGE
```

The second marker always locates the last quantum of the dimension, either directly, by specifying a negatively-signed axis marker, or indirectly, by specifying a positively-signed axis marker that counts forward from marker1.

NOTE 135 In the former case, the quantum count can be calculated from the positions of the first and last quanta (last-first+1); in the latter case, the quantum count is the value of the second marker.

NOTE 136 Coordinate addressing can be undependable if the addressed object is subject to modification, as quanta can be removed and added. To visualize this easily, think of a paragraph as an axis and the characters in it as quanta. To address the second sentence, which currently contains characters 21 through 30, the dimension would be “21 10”. With this form of address, you would get extraneous characters if the sentence were later shortened.

To minimize such problems, either or both of the axis markers used to specify the dimension can do so with respect to either end of the range. If the paragraph in the example had 40 characters, the second sentence could also be addressed as “21 -11”. With the dimension expressed in this way, the second sentence would be addressed in full regardless of how it was later shortened or lengthened. (Provided, of course, that the surrounding text was not modified.)

The effect of using signed axis markers can be summarized as follows:

1 1	means first quantum in the range
-1 1	means last quantum in the range
1 -1	means 1st to last (entire range)
1 -2	means all but last quantum
2 -1	means all but first quantum
5 6	means 5th through 10th quanta, inclusive

When the second marker is negative, a negative first marker has a slightly different meaning. It still counts from the *direction* of the end of the range, but from the second marker, rather than the last quantum.

For example, in a range of ten quanta, “-3 -2” would mean the 7th through 9th quanta, inclusive. The first marker is the negation of the quantum count.

```
        <!-- HyTime Dimension Specification Notation -->
<![ %HyDimSpc; [
<!notation
   HyDimSpc       -- HyTime Dimension Specification Notation --
                  -- Clause: 6.8.2 --
                  -- A pair of axis markers (in the HyTime Axis Marker
                     List Notation, after resolution of marker
                     functions) that together specify a position and
                     quantum count along a single axis. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Dimension Specification Notation//EN"

-- Attributes [base]: HyDimSpc --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDimSpc       -- HyTime Dimension Specification Notation --
                  -- Clause: 6.8.2 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyMrkLst
>
<!entity % HyMrkLst "INCLUDE">
]]><!-- HyDimSpc -->
```

6.8.3 Dimension Specification

The element form **dimension specification** (*dimspec*) contains elements or data that are interpreted as a single dimension on a single axis.

```
                   <!-- Dimension specification -->
<![ %dimspec; [
<![ %HyDimSpc; [
   <!entity % ddimspec "HyDimSpc">
]]>
<!entity %
```

```
   ddimspec       -- Default dimension specification notation --
                  -- Clause: 6.8.3 --

   "#REQUIRED"
>
<!element
   dimspec        -- Dimension specification --
                  -- Clause: 6.8.3 --
                  -- A position and quantum count along a single
                     axis. When used as a marker function, returns a
                     pair of positive axis markers, the first being
                     the position of the first quantum of the
                     dimension and the second being the quantum count
                     of the dimension. --
   O O
   (%HyCFC;|%marklist;)*
                  -- Constraint: Content must conform to the notation
                     specified in the dimspec's notation attribute. --

-- Attributes [base]: dimspec --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dimspec        -- Dimension specification --
                  -- Clause: 6.8.3 --

   HyBase   NAME     #FIXED markfun

   notation       -- Dimension specification notation --
      NAME        -- Lextype: NOTATION --
      %ddimspec;
>
]]><!-- dimspec -->
```

6.8.4 Dimension List

The notation form **HyTime dimension list notation** (*HyDimLst*) allows the specification of lists of dimensions without requiring explicit dimension specification elements.

```
               <!-- HyTime Dimension List Notation -->
<![ %HyDimLst; [
<!notation
   HyDimLst       -- HyTime Dimension List Notation --
                  -- Clause: 6.8.4 --
                  -- A list of dimensions each of which may be
                     specified as either a pair of axis markers (in
                     the HyTime Marker List Notation), or as all or
                     part of the list of axis markers returned by a
                     marker function. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Dimension List Notation//EN"
-- Attributes [base]: HyDimLst --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDimLst       -- HyTime Dimension List Notation --
                  -- Clause: 6.8.4 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyMrkLst
>
<!entity % HyMrkLst "INCLUDE">
]]><!-- HyDimLst -->



<!entity %
   dimlist        -- HyTime Dimension List content --
                  -- Clause: 6.8.4 --
   "%marklist;|dimspec"
>
```

6.8.5 Overrun handling

The attribute form *overrun* is associated with element forms that specify coordinate locations.

The attribute **overrun handling** (*overrun*) specifies how an actual dimension that overruns the limits of the addressable range will be handled. It will either be treated as an error (“error”), wrapped around modulo the length of the range (“wrap”), or truncated to equal the first and/or last quantum of the range, as required (“trunc”). If “trunc” is specified:

— If some part of the dimension occurs within the range, and either marker is before the first quantum of the range, it is treated as the first quantum.

— If some part of the dimension occurs within the range, and either marker is after the last quantum of the range, it is treated as the last quantum.

— If no part of the dimension occurs within the range, a specification of “trunc” is equivalent to a specification of “ignore”.

Alternatively, “ignore” can be specified, in which case the dimension will be treated as though it had not been specified. In some circumstances, an error could result.

```
                      <!-- Overrun Handling -->
<![ %overrun; [
<!attlist
-- overrun --     -- Overrun handling --
                  -- Clause: 6.8.5 --
   (dataloc,dimref,event,evgrp,fcsloc,listloc,modgrp,modscope,
    pathloc,progrp,proscope,relloc,treeloc)

   overrun        -- Overrun Handling --
                  -- Handling of dimension that overruns range --
      (error|ignore|trunc|wrap)
      error
>
]]><!-- overrun -->
```

6.8.6 HyTime Marker Function Language (HyFunk)

The **HyTime Marker Function Language** (*HyFunk*) provides the minimum functions needed to perform integer arithmetic on markers in marker functions. It is based on the DSSSL Expression Language.

```
               <!-- HyTime Marker Function Language -->
<![ %HyFunk; [
<!notation
   HyFunk         -- HyTime Marker Function Language --
                  -- Clause: 6.8.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Marker Function Language//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
]]><!-- HyFunk -->
```

HyFunk is a Scheme-style function language consisting of the arithmetic functions addition, multiplication, subtraction, quotient, remainder, and modulo as well as a function for performing dimension references. The arithmetic functions are identical to the corresponding procedures in the DSSSL expression language. The dimref function emulates the dimension reference element form (see *9.8 Dimension referencing.*). HyFunk functions always return a single integer.

In DSSSL, functions are lists, enclosed in parentheses, where the function name is the first member of the list. Lists are s-separator delimited. The arguments to the HyFunk arithmetic functions are:

[1] arg =
(*marker* **|** *HyFunk function*)

[2] marker =
snzi

Where *HyFunk function* is one of the following arithmetic functions or the dimref function:

Addition — Returns the sum of its arguments.

[3] addition =
"(+", (*arg*, *arg*+), ")"

Multiplication — Returns the product of its arguments.

[4] multiplication =
"(*", (*arg*, *arg*+), ")"

Subtraction — With two or more arguments, returns the difference of its arguments, associating to the left; with one argument, returns the negation of its argument.

[5] subtraction =
"(-", (*arg*, *arg*+), ")"

Number-theoretic division

These functions implement number-theoretic (integer) division: For positive integers n1 and n2, if n3 and n4 are integers such that n1 = n2n3 + n4 and 0 <= n4 < n2, then the following is true.

```
(quotient n1 n2) -> n3
(remainder n1 n2)-> n4
(modulo n1 n2) -> n4
```

[6] quotient =
"(quotient", (*arg*, *arg*), ")"

[7] remainder =
"(remainder", (*arg*, *arg*), ")"

[8] modulo =
"(modulo", (*arg*, *arg*), ")"

The dimref function returns the specified dimension component of the referenced element. The parameters of the dimref function correspond to the attributes of the dimref element form and have the same meaning and constraints (see *9.8 Dimension referencing*).

[9] dimref =
"(dimref", **IDREF**,
("selcomp: (" "'first" | "'last" | "'qcnt" |
"'precedng" | "'followng")),
("flip", ("#t" | "#f"))?,
("schdspec:" **IDREF**)?,
("axisspec:" **IDREF**)?,
("dimnum:" *number*)?,
("extnum:" *number*)?,
("granule:" *granule*)?,
("roffgran: (" "'roerror" | "'rocmplet" | "'roscant" | "'roboundy"))
("dsdtypes:" **string**)?,
("prjdirct: (" "'drctonly" | "'drctproj" | "'projonly"))
("prjtarg:" **IDREF**)?,
("prjby:" **IDREF**)?)

HyFunk conforms to all the requirements of and restrictions on the corresponding DSSSL procedures defined in ISO/IEC 10179:1996 Document Style Semantics and Specification Language.

7 Location address module

This clause describes the optional location address module of HyTime.

NOTE 137 In addition to the other optional facilities of the location address module, each location address element form is itself an optional facility.

7.1 Concepts and definitions

The fundamental form of location is that which coincides with the fundamental form of addressing; that is, an element in the current document that is identified by an ID. This form of location is supported directly by SGML. It can be addressed by attributes of the type “ID reference” (IDREF) and “ID reference list” (IDREFS).

However, SGML provides no direct way to assign an ID to arbitrary pieces of data, to elements in read-only documents that have no IDs, to collections of elements, and to other arbitrary locations.

As elements need not have unique identifiers, and as hypermedia linking need not be to complete elements, the HyTime location address module provides the ability to define “location address” elements that associate an ID with an object at an arbitrary location (that is, a location addressed by some form of address other than a local ID). When an ID reference is made to the ID of a location address, it is interpreted as a reference to the located object.

NOTE 138 When a location address element is simply encountered in the content of a document, however, it is not automatically resolved. Its attributes and data are passed to the application by the SGML parser in the normal way. The application could, as part of its processing, request the HyTime engine to resolve the location address, but there is no requirement that it do so.

NOTE 139 With few exceptions, HyTime does not distinguish locations by the time needed to access them in storage, as the time could vary depending on the environment in which the document is processed. In practice, a HyTime engine could alert an application when access is expected to exceed a previously specified threshold. The HyTime BOS control facilities can be used to organize the members of a hyperdocument so as to allow optimization of object access based on expected access time.

Location addresses operate on groves and always return nodes in groves (see *7.1.4 Groves and Location Addressing*).

NOTE 140 Informally it is normal to speak of “locating an element”; formally this means “locating the node derived from the element in the applicable grove.”

7.1.1 Location types

The following list summarizes the types of object that can be addressed in a HyTime document, categorized by the form of address.

— Nodes addressed by a name (“name-space locations”)

- Entity (uses either direct entity reference by attributes with a value prescription of ENTITY or “name-space location address”).
- Identified external element: element with an ID in another document (uses “name-space location address”).
- Unidentified document element: the root element of this or another document that has no ID (uses “named location address”).
- Identified local element: an element in this document that is not a location address but has an ID (does not use a location address).
- Property value: the value of a property of a node in a grove addressed by property name, such as the value property of an attribute node in an SGML document grove (uses “property location address”).

— Locations addressed by a coordinate position (“coordinate locations”)

- List nodes: nodes in a node list addressed by position within the list (uses “list location address”).
- Tree nodes: nodes in a tree addressed by their positions within a tree (uses “tree location address”), by their positions within a table of paths from the tree root to the leaves (uses “path location address”), or by their genealogical relationshipa to other nodes (uses “relative location address”).
- Scheduled objects: objects occurring within a region of a finite coordinate space. Addressed either by coordinate location (uses “FCS location address”) or by absolute date and time (requires the “calendar

specification” option, see *9.9.2 Calendar specification*). Requires the scheduling module (see *9.10 Finite coordinate space location address*).

— Objects addressed by a semantic construct (“semantic locations”)

- Nodes addressed by queries on their properties (uses “query location address”).
- Inaccessible (by computer) object: description of a document, person, or other information object to which access cannot be automated (uses “bibliographic location address”).

NOTE 141 In terms of the theory of measurement, name-space locations are positions on a “nominal” scale. Coordinate locations are positions on either an “ordinal”, “interval”, or “ratio” scale, depending on how the application chooses to construct the axes and schedules, the SMU and granules used, how dimensions are specified, and any semantics attached to the element types or granule names by the application. For example, an ordinal scale can be represented by creating a resource pool of extents corresponding to the intervals of the scale, and positioning events by means of dimension references to those extents (see note 279 in *9.4 Scheduling and extents*).

A location address element can address more than one object simultaneously, creating a “multiple location”. No significance is attached to the grouping implied by such a “multiple location address”: the locations are treated as though each had been referenced individually. However, the elements that use multiple location addresses may associate specific semantics with multiple locations. Multiple locations require support for the “multloc” option.

NOTE 142 Attributes with a declared value of “IDREFS” do not constitute multiple locations unless they are also given a location type of “IDLOC” through the use of the refloc facility (see *7.8 Reference location address*).

7.1.2 Location Sources

Location addresses operate by selecting nodes from a group of nodes. The set of nodes from which a location address selects is its “location source”. The location source for one location address may be the nodes selected by another location address. The second location address is the location source of the first location address.

Because location addresses operate on nodes in a grove, the ultimate location source for every location address is the grove or groves that contain the nodes addressed by the location address. Groves are addressed by addressing the document or data entities from which they are constructed or by addressing grove definition elements. Groves may also be addressed implicitly by semantics defined for particular location address elements. Addressing a grove is equivalent to addressing the root node of the grove (the “grove root”).

NOTE 143 The grove formalism used by HyTime is needed for definitional purposes but need not be exposed to users and document authors under normal circumstances. The location addressing mechanisms have been designed so that their default behavior will almost always be what an author would intuitively expect.

When one location address acts as the location source for another location address, a “location ladder” is formed. Each “rung” in the location ladder forms a location source for the rung below it. The top rung of the location ladder always has a grove or groves as its location source. The bottom rung of a location ladder is the location address to which initial reference is made. The resolution of a location ladder in which a treeloc is the bottom rung and a nmsploc is the top rung can be described as follows::

1) A contextual link addresses a tree location address element (treeloc) by its ID.
2) The treeloc refers to a name-space location address (nmsploc) as its location source.
3) The nmsploc names another document entity as its location source (implicitly addressing the value of the elements property of the SGML document grove derived from the document entity).
4) The nmsploc addresses an element within the document by selecting it by name from the element nodes listed in the value of the elements property.
5) The element node selected by the nmsploc is the root of the tree the treeloc selects from.
6) The treeloc selects a node in the tree by specifying a tree position.

NOTE 144 This location ladder can be viewed schematically as:

```
         Document
         Entity --->SGMLDOC node (exhibits "elements"
            A       name space for nmsploc)
            |
            |
         nmsploc--->Element addressed by name
            A       (becomes root of tree for treeloc)
            |
            |
Clink--->Treeloc--->Element in tree (anchor of clink)
```

The rungs of the location ladders needed to address groves may be implied under the circumstances defined for each form of location address. For example, the implicit grove address for name-space locations described above can be made explicit by creating a grove definition whose “grove source” is the document entity. That grove definition is then the location source for a property location address that addresses the elements property of the grove root. This property location address would then be the location source for the name-space location address.

NOTE 145 Implicit grove addressing is conceptually a form of markup minimization because any implicit grove addressing behavior can be replaced by the equivalent direct or indirect reference to a grove definition. For example, the default implied location source for name-space location addresses is the “elements” property of the SGMLDOC object in an SGML document grove. This location ladder could be specified explicitly as follows:

```
<!-- Note: Entity DocB declared as data entity with notation SGML -->
<pgrovdef id=docbgrov grovesrc=DocB grovplan=HyTime>
<nmsploc id=docb.foo namespc=elements locsrc=docbgrov>foo</nmsploc>
```

Each of the implied location sources can be specified explicitly with the appropriate location ladder. In general, any node in a grove can be located by a combination of property locations, name-space locations, list locations, and tree locations whose ultimate location source is an explicit or implicit grove constructor.

This International Standard does not require an implementation to reflect the discrete rungs of a location ladder, or to pass information between those rungs in the form specified in a HyTime document.

NOTE 146 This provision allows a HyTime engine to optimize interpretation of a location ladder and access to the bottom rung.

NOTE 147 The HyTime constructs for location addressing and location ladder construction are quite robust. They are based on a few simple rules that allow modular implementation of a HyTime engine. As a result, it is possible to construct location ladders that could in some contexts appear to be unusual, but which HyTime would accept as valid and process consistently.

For example, in an SGML parsing context, a book chapter can be used as the location source for dataloc, even though the chapter could contain subelements as well as data. HyTime, following the rules for recognizing word tokens as quanta, would tokenize all of the data in the content of the chapter and its subelements, which could be useful if an application wanted to count the words, for example.

Restricting the generality of location addressing would introduce needless policy decisions into HyTime. It is for the applications using HyTime to decide whether to restrict any forms of location address; there is no way for HyTime to determine what one application might consider to be nonsense and another would find to be useful, or even essential.

7.1.3 Location Paths

Objects that do not have names can be given names by location address elements that have IDs and that address the objects, directly or indirectly.

For example, a tree location address element with an ID associates its ID with the node it addresses such that a reference to the tree location address' ID is equivalent to having used that ID on the node directly.

Thus, when a location address element addresses another location address element, the effect is to address whatever the second location address addresses, regardless of how the first location address addresses the second location address. For example, a tree location address that happens to address a name-space location will effectively address the objects addressed by the name-space location.

NOTE 148 The only exception to this is when reference is made to a location address element by a referential attribute or content that has an associated reference range of "D" (direct reference). See *7.7.2 Reference resolution range*.

"Location path" is the term for the indirect address chaining that occurs when one location element addresses another. Each is a step in the location path. The locsrc of a path step is the bottom rung (and possibly the only rung, in which case it is also the top rung) of its location ladder.

Note that the top rung of a location ladder is always a grove root, represented by a grove definition element or the entity from which the grove is constructed. Normally, the upper rungs of the ladder are implied from the specified locsrc. Only location address elements can be part of a location path or location ladder.

NOTE 149 Visually, paths go from left to right; ladders are above the steps that they are the location source of. For example, a two-step path using two name-space location addresses, each addressing element IDs, the first addressing the second, can be viewed schematically as:

```
         Document     Document Entity
         Entity       (contains nodes
         (contains    addressed by
         nmsploc[2])  nmsploc[2])
           A                A
           |                |
           |                |
Clink--->nmsploc[1]--->nmsploc[2]--->nodes addressed by name (anchor of clink)
```

Each name-space location is a step in the path from the clink (the referrer) to the nodes ultimately addressed (which become the anchor of the clink). Above each name-space location is its location ladder. For the first name-space location, the location source is the document entity that contains the second name-space location address. For the second name-space location, the location source is the document entity that contains the nodes it names.

NOTE 150 The fact that different path steps in different paths might use the same elements as rungs of their location ladders is irrelevant from an information modeling standpoint, but might be of interest to an implementation. The same is true for common elements used in different paths. In other words, with respect to a given location address element, HyTime knows nothing of what is below or to the left of it; it knows only the rungs of its location ladder above (locsrc and its locsrc, etc.) and the steps to the right (the located object and the object it locates, etc.).

When the multloc option is not supported, a location path is always a single sequence of steps connecting a single initial reference to a single object that is not a location address. When the multloc option is supported, any step in a path can address multiple location addresses as well as non-location address elements, creating a "fan out" of the path. However, the end result of resolving a branching location path is a single list of nodes.

NOTE 151 It is an RHE for a step in a location path to address itself directly or indirectly.

Each step in a location path has its own location ladder.

7.1.4 Groves and Location Addressing

All location addresses operate on and address nodes in groves. Groves are constructed by "grove construction processes" according to a particular property set and grove plan. The ultimate location source for any location ladder is a grove. In HyTime, groves may be addressed implicitly by the rules for implied location sources and the semantics of specialized location address elements. Groves may also be defined and addressed explicitly. In addition, any location address element can use a grove plan different from that used to construct the grove that is its ultimate location source in order to address the grove according to a particular "view".

See *A.4 Property Set Definition Requirements (PSDR)* for more information on groves in general.

7.1.4.1 Grove Plan

A grove is constructed according to a particular property set, which defines the set of node classes and properties from which a grove is constructed. Each data notation must have its own property set (e.g., the SGML Property Set for the SGML notation, see *A.7 SGML Property Set*). Which classes and properties are included in the constructed grove is determined by the “grove plan” used by the grove constructor. The grove plan is either specified explicitly or is a default grove plan, defined either in the property set itself or through some other defaulting mechanism. The grove plan that includes all classes and properties in the property set is the “complete grove plan” and results in a “complete grove.” Property sets typically define a default grove plan that is not a complete grove plan.

Grove plans are specified by referring to grove plan elements from grove definition elements, data entities, and location address elements. Explicit grove plans and references to them require support for the grovplan option.

Grove plans can be used to control how a grove is viewed by a particular location address. When applied to an already constructed grove, a grove plan acts as a “filter” on the grove for the purposes of resolving the location address.

NOTE 152 The grove plan does not modify the grove from which the addressed nodes are selected. For example, properties of the addressed nodes excluded by the grove plan are still available.

The element form **grove plan** (grovplan) defines a grove plan for use by location address elements or by a HyTime document (to define the grove plan used to construct a HyTime document's effective SGML grove, see *6.4 HyTime document*). A grove plan contains subelements that list the modules, classes, and properties to be included or omitted when constructing or addressing a grove. A grove plan modifies the default grove plan defined in the property set to which the grove plan refers.

The attribute **property set** (*propset*) refers to the entity containing the property set definition document to which the grove plan applies.

NOTE 153 It is not necessarily an error if the property set definition document cannot be accessed; an implementation might not require access to it as knowledge of the property set is normally built into the relevant software.

The element forms **included modules** (*inclmods*) and **omitted modules** (*omitmods*) list modules to be included or omitted. Including or omitting a module does not preclude separately including or omitting classes or properties defined in that module. The omission of a module always supersedes inclusion of the module in the same grove plan. The content of an inclmods or omitmods element must be a list of module names.

The element forms **included classes** (*inclclas*) and **omitted classes** (*omitclas*) list classes to be included or omitted. The omission of a class always supersedes inclusion of the class. The content of an inclclas or omitclas element must be a list of class names.

The element forms **included properties** (*inclprop*) and **omitted properties** (*omitprop*) list properties to be included or omitted. The attribute **classes from which to include or omit** (*classes*) lists the classes from which the property is to be included or omitted. If no value is specified, the properties are included or omitted from all classes that exhibit the property. The omission of a class always supersedes the inclusion of any of its properties. The omission of a property always supersedes inclusion of the property. The content of an inclprop or omitprop element must be a list of property names.

```
                        <!-- Grove Plan -->
<![ %grovplan; [
<!element
   grovplan       -- Grove plan --
                  -- Clause: 7.1.4.1 --
                  -- Specifies a subset of a complete grove's address
                     space. --
   - O
   (inclmod?,omitmod?,inclclas?,omitclas?,inclprop?,omitprop?)
```

```
                  -- Constraint: Can include classes and properties
                     from omitted modules (but not properties from
                     omitted classes). --

-- Attributes [locs]: grovplan --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [locs]: agrovdef:grovplan, dgrvplan:grovplan,
   egrovplan:grovplan, lgrvplan:grovplan, pgrovdef:grovplan --
>
<!attlist
   grovplan       -- Grove plan --
                  -- Clause: 7.1.4.1 --

   propset        -- Property set definition document --
      ENTITY      -- Constraint: not an error if the property
                     set document entity is not available. --
      #REQUIRED
>
<!element
-- inomcomp --    -- Include component/omit component elements --
                  -- Clause: 7.1.4.1 --
   (inclclas,inclmod,omitclas,omitmod)

   - O
   (#PCDATA)      -- Lextype: cnmlist --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!element
-- inomprop --    -- Include property/omit property --
                  -- Clause: 7.1.4.1 --
   (inclprop,omitprop)

   - O
   (#PCDATA)      -- Lextype: cnmlist --

-- Attributes [locs]: inomprop --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
-- inomprop --    -- Include property/omit property --
                  -- Clause: 7.1.4.1 --
   (inclprop,omitprop)

   classes        -- Classes from which to include or omit the named
                     properties --
      NAMES
      #IMPLIED    -- Default: all classes that exhibit the
                     property. --
>
]]><!-- grovplan -->
```

For HyTime document elements, the attribute **grove plan** (*grovplan*) refers to the grovplan element to be used to control the construction of the HyTime extended SGML document grove for this document. The grove plan element can be in another document if external referencing is supported.

```
                 <!-- HyTime Document Grove Plan -->
<![ %grovplan; [
<!attlist
-- dgrvplan --    -- HyTime document grove plan --
                  -- Clause: 7.1.4.1 --
   (HyDoc)

   grovplan       -- Grove plan --
                  -- Grove plan for HyTime extended SGML document
                     grove --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: HyTime default grove plan --
>
]]><!-- grovplan -->
```

For data entities named as location sources, the data attribute **grove plan** (*grovplan*) refers to the grovplan element to be used to control the construction of the principal grove for an entity. The value is interpreted as a direct or indirect ID reference to a grove plan. The grove plan element can be in another document if external referencing is supported. If not specified, the default grove plan for the entity's data content notation is used.

```
                      <!-- Entity Grove Plan -->
<![ %grovplan; [
<!attlist #NOTATION
-- egrvplan --    -- Entity grove plan --
                  -- Clause: 7.1.4.1 --
   #ALL

   grovplan       -- Grove plan --
                  -- Grove plan to use when constructing grove from
                     entity. --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: default grove plan for entity's DCN. --
>
]]><!-- grovplan -->
```

For location address elements, the attribute **grove plan** (*grovplan*) names the grove plans to be used to interpret the grove that is the direct or indirect location source (*7.2 Location source*). It is not an error to refer to grove plans for property sets that are not used by the location source.

```
<![ %grovplan; [
<!attlist
-- lgrvplan --    -- Location source grove plan --
                  -- Clause: 7.1.4.1 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmquery,nmsploc,
    pathloc,proploc,queryloc,relloc,treeloc)
    grovplan      -- Grove plans --
                  -- Grove plans that govern resolution of address --
       CDATA      -- Reference --
                  -- Reftype: grovplan* --
       #IMPLIED   -- Default: grove plans of location source groves as
                     defined by their grove definitions, whether
                     explicitly specified or implied. --
>
]]><!-- grovplan -->
```

NOTE 154 Indirect addressing may be used to refer to grove plans in a document other than the one in which the location address, grove definition, or HyTime document element that uses it occurs.

7.1.4.2 HyTime Default SGML Grove Plan

As defined by the SGML property set, the principal tree in an SGML document is the tree of elements and their content rooted at the document element. The SGML property set allows everything that can occur within an element to be in an element's content property (and thus be a child of the element for the purpose of tree addressing). However, because HyTime is intended to be used primarily to address elements and their “semantic content” (informally, the content of the element that would normally be presented in a formatted view of the document), HyTime defines a default view of SGML documents that restricts the children of elements to elements and data portions.

The HyTime default grove plan restricts the content of elements to two node types: elements and pseudo-elements. Pseudo-elements represent a contiguous sequence of nodes allowed in mixed content between a tag close and the next tag open.

NOTE 155 Pseudo-elements make it easier to address elements and character data using tree location addresses by making tree positions insensitive to the number of data characters between elements (except in the case of addition or deletion of complete pseudo-elements).

Because they are not considered children of elements in the HyTime default grove plan, comments, markup declarations, and processing instructions are ignored when determining the content of pseudo-element nodes. (But these node classes will be in the content of pseudo-elements returned by HyTime location addresses if the grove was constructed with a grove plan that includes these node classes generally.)

Location addresses default to using the grove plan of the constructed grove. The HyTime default grove plan is used to construct the HyTime effective SGML grove for HyTime documents for which an explicit grove plan is not specified (either through the grovplan attribute of the hydoc element form or the grovplan data attribute for SGML document entities). The HyTime grove plan modifies the SGML default grove plan as defined by the SGML property set (the SGML default grove plan is also the DSSSL default grove plan).

The HyTime default grove plan is:

```
<!DOCTYPE grovplan [

<?IS10744 ArcBase HyTime>
<!NOTATION HyTime
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!ATTLIST #NOTATION HyTime
   ArcDocF  NAME     #FIXED grovplan
   ArcDataF NAME     #FIXED HyBridN
   ArcDTD   CDATA    #FIXED "HyTime"
   ArcQuant CDATA    #FIXED "NAMELEN 9"
   ArcOpt   CDATA    #FIXED "grovplan"
>
<!NOTATION AFDRMeta
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD Notation//EN"
>
<!ENTITY HyTime
   PUBLIC "ISO/IEC 10744:1997//DTD AFDR Meta-DTD
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
   CDATA AFDRMeta
>

<!ELEMENT grovplan
   - - (title,desc,inclmod?,omitmod?,inclclas?,omitclas?)
>
<!ATTLIST grovplan
   id       ID       #IMPLIED
   propset  ENTITY   #REQUIRED
>
<!ELEMENT (title,desc,inclclas,inclmod,omitclas,omitmod)
   - - (#PCDATA)
>

<!NOTATION PROPSET
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Property Set Definition Architecture//EN"
>
<!ENTITY SGMLProp
   PUBLIC "ISO/IEC 10744:1997//DOCUMENT SGML Property Set//EN"
   CDATA PROPSET
>
]>
<grovplan propset=SGMLProp id=htdefgp>
<title>HyTime Default SGML Grove Plan</title>
<desc>
Removes processing instructions (pi) from and
adds pseudo-elements (pelement) to the default SGML
grove plan defined in the SGML property set.
</desc><inclmod>
pelement
</inclmod><omitclas>
pi
</omitclas>
</grovplan>
```

7.1.4.3 Effective SGML Document Grove Plan

When a HyTime engine processes a HyTime document it produces an “effective SGML document grove” that includes classes and properties unique to HyTime. By default the effective SGML grove is constructed using the HyTime default grove plan. However, the HyTime document element can name a different grove plan to use when constructing the effective SGML grove for the document using the grovplan attribute of the hydoc element form (see *7.1.4.1 Grove Plan*).

7.1.4.4 Grove Definition Elements

The ultimate location source for any location address is a grove, represented by the grove root. The grove definition element forms enable explicit references to groves. Reference to a grove definition causes the construction of the grove if it has not already been constructed. A grove definition associates source data (the “grove source”) with a grove construction process definition and, optionally, a grove plan.

A HyTime engine constructs only one grove for each combination of grove source and grove construction process. If two or more grove definitions share both grove source and grove construction process, the grove plan of the constructed grove is the union of the sets of classes and properties defined by the grove plans of the grove definitions. The grove plans specified by each grove definition then act as filters over the constructed grove, just as for grove plans specified by location address elements, but applying to all further processing based on that particular grove definition.

The element form **primary grove definition** (*pgrovdef*) defines groves constructed from data that is not already in a grove (for example, an unparsed SGML document). The attribute **grove source** (*grovesrc*) refers to the entity that contains the source data. The attribute **grove construction process** (*grovecon*) names the grove construction process to be used. If grovecon is not specified, the grove construction process is defined by the notation of the grove source entity.

NOTE 156 The values of attributes defined for the grove construction process may be specified through use of the DAFE facility of the General Architecture (see *A.5.3 Data Attributes for Elements (DAFE).*)

The attribute **grove plan** (*grovplan*) refers to the grove plan that controls the construction of the grove. If grovplan is not specified, the grove plan is the default grove plan used by the grove construction process (which may be the default grove plan defined by the property set for the source entity's notation).

The element form **auxiliary grove definition** (*agrovdef*) defines groves constructed from nodes in an existing grove (for example, an SGML document grove). The attribute **grove source** (*grovesrc*) addresses, directly or indirectly, the nodes from which the auxiliary grove is constructed. The attribute **grove construction process** (*grovecon*) names the grove construction process to be used. A grove construction process must be specified for auxiliary groves. The attribute **grove plan** (*grovplan*) refers to the grove plan that controls the construction of the grove. If grovplan is not specified, the grove plan is the default grove plan used by the grove construction process.

Support for the grovedef option allows both pgrovdef and agrovdef to be used.

```
                  <!-- Primary Grove Definition -->
<![ %pgrovdef; [
<!element
   pgrovdef       -- Primary grove definition --
                  -- Clause: 7.1.4.4 --
   - O
   (%HyCFC;)*

-- Attributes [locs]: pgrovdef --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   pgrovdef       -- Primary grove definition --
                  -- Clause: 7.1.4.4 --

   grovesrc       -- Grove source --
      ENTITY
      #REQUIRED

   grovecon       -- Grove construction process --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: inherent in notation of grove source --

   grovplan       -- Grove plan --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: default grove plan of grove
                     construction process. --
>
]]><!-- pgrovdef -->

                 <!-- Auxiliary Grove Definition -->
<![ %agrovdef; [
<!element
   agrovdef       -- Auxiliary grove definition --
                  -- Clause: 7.1.4.4 --
   - O
   (%HyCFC;)*

-- Attributes [locs]: agrovdef --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   agrovdef       -- Auxiliary grove definition --
                  -- Clause: 7.1.4.4 --

   grovesrc       -- Grove source --
      CDATA       -- Reference --
      #REQUIRED

   grovecon       -- Grove construction process --
      NAME        -- Lextype: NOTATION --
      #REQUIRED

   grovplan       -- Grove plan --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: default grove plan of grove
                     construction process. --
>
]]><!-- agrovdef -->
```

7.2 Location source

The attribute form **location source** (*locsrc*) is used by all location address element forms. All location addresses address nodes by selecting them from a group of nodes addressed by another location address or by direct or implicit reference to a grove or groves. The attribute **location source** (*locsrc*) addresses, directly or indirectly, the location source for a location address. When explicit grove plans are supported, location address elements can refer to the grove plans to use for the location source groves (see *7.1.4.1 Grove Plan*).

When a location source is omitted or the location source is a reference to a document, subdocument, or data entity, the location source is implied. The implied location source is specified by the **implied location source** attributes (see *7.3 Implied location source*).

When the location source is a grove definition element, the location source is the grove root of the grove constructed as defined by the grove definition (see *7.1.4.4 Grove Definition Elements*).

The attribute **cannot construct grove** (*cantcnst*) indicates the expected system response when the system is unable to construct primary groves from data addressed as the location source: either report an RHE (“error”) or ignore the location address (“ignore”).

```
                       <!-- Location Source -->
<![ %locsrc; [
<!attlist
-- locsrc --      -- Location source attributes --
                  -- Clause: 7.2 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmquery,nmsploc,
    pathloc,proploc,queryloc,relloc,treeloc)

   locsrc         -- location source --
      CDATA       -- Reference --
                  -- Constraint: References to entities are references
                     to roots of primary groves constructed from the
                     entities. --
      #IMPLIED

   cantcnst       -- Cannot construct grove from located data --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
]]><!-- locsrc -->
```

7.3 Implied location source

A location source is implied when either the locsrc attribute is not specified at all or when the locsrc attribute names a document, subdocument, or data entity. If the locsrc attribute is completely omitted, the location source is within the SGML grove constructed from the document that contains the location address or is specified by an attribute of the referring element. If the location source is a document, subdocument, or data entity, the location source is within the grove constructed from that entity.

The attribute form **implied location source** (*impsrc*) specifies the implied location source for location addresses. The implied location source can be the principal tree root of the grove (ptreert) (the document element for SGML document groves), the grove root (grovert), the non-location address element that referred (directly or indirectly) to the location ladder of which the location address element is the top rung (referrer), or the location source addressed by a referential attribute of the referrer element (referatt). For query location addresses, the location source can be implicit in the query itself (implicit). For other location addresses, the value “implicit” is synonymous with “ptreert”. Note that only the top run of a location ladder can have an implied location source.

The attribute **location source attribute of referrer** (*referatt*) names the attribute of the referrer that addresses the location source for the location address. The attribute or content named must be exhibited by the referrer and must be referential.

NOTE 157 When the referrer or referatt default location source is used, the same location address element may address different nodes when used by different referrers.

```
            <!-- Implied Location Source -->
<![ %impsrc; [
<!attlist
-- impsrc --      -- Implied location source --
                  -- Clause: 7.3 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmsploc,nmquery,
    pathloc,proploc,queryloc,relloc,treeloc)

   impsrc         -- Implied location source --
                  -- Default location source for implied locsrc
                     attribute.  Possible values are:

                     ptreert   Default location source is node in
                               default grove that serves as principal
                               tree root, normally the document
                               element.

                     grovert   Grove root of document that contains
                               location address.

                     referrer  Default location source is non-location
                               address object making direct or
                               indirect reference to the location
                               ladder of which this location address
                               is the top rung.

                     referatt  Default location source is attribute
                               of referrer named by referatt
                               attribute.
                     implicit  Location source is implicit in query.
                               For queryloc, location source is
                               implicit in query.  For other location
                               address forms, same as ptreert.
                     --
      (grovert|implicit|ptreert|referatt|referrer)
                  -- Constraint: referatt allowed only when referatt
                     option is supported. --
      ptreert
>
]]><!-- impsrc -->


    <!-- Attribute Of Referrer That Addresses Location Source -->
<![ %referatt; [
<!attlist
-- referatt --    -- Attribute of referrer that addresses location
                     source --
                  -- Clause: 7.3 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmquery,nmsploc,
    pathloc,proploc,queryloc,relloc,treeloc)

   referatt       -- Attribute of referrer that addresses location
                     source --
      CDATA       -- Lextype: ATTORCON --
      #IMPLIED    -- Constraint: named attribute or content must be
                     exhibited by referrer and must be referential. --
>
]]><!-- referatt -->
```

7.4 Multiple location

The attribute form *multloc* allows the properties of ordering and duplicate omission to be associated with a multiple location address element.

If multloc attributes are specified for an element that is not a multiple, it is not an RHE, but the attributes will be ignored by the HyTime engine.

A multiple location is inherently ordered. The attribute **location ordering** (*ordering*) indicates whether the order of the multiple locations is significant to the application ("ordered") or not ("noorder").

NOTE 158 This attribute might allow a HyTime engine to optimize access to the individual locations.

The attribute **set** (*set*) treats the multiple locations as an ordered set by ignoring any duplicate nodes in the resulting node list.

```
                <!-- Multiple Location Attributes -->
<![ %multloc; [
<!attlist
-- multloc --     -- Multiple location attributes --
                  -- Clause: 7.4 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,mixedloc,nameloc,nmlist,
    nmquery,nmsploc,pathloc,proploc,queryloc,relloc,treeloc)

   ordering       -- Is ordering of locations significant? --
      (noorder|ordered)
      ordered

   set            -- Make multiple a set by ignoring duplicates --
      (notset|set)
      notset
>
]]><!-- multloc -->
```

7.5 Tree type

Groves define two distinct types of trees: content trees and subnode trees. Content trees are formed by the content properties of nodes such that a node that exhibits a content property is the parent of the nodes in the property's value. Subnode trees are formed by the subnode properties of nodes such that a node is said to be in the subnode tree rooted at another node if that node is the origin of the first node.

NOTE 159 The tree of elements and their content in an SGML document grove is a content tree. The tree of elements and all their subnodal properties (attributes, element type, as well as content) is a subnode tree.

The attribute **tree type** (*treetype*) controls whether the tree used by a location address as its location source is a content tree ("content") or a subnode tree ("subnode").

NOTE 160 When a location address uses a grove root as a subnode tree it addresses every node in the grove as its location source.

The treetype attribute is ignored for listloc, nameloc, nmlist, nmquery, nmsploc, queryloc, and dataloc unless they are addressing spans.

```
                     <!-- Tree type attribute -->
<![ %treetype; [
<![ %spanloc; [
   <!entity % treetpel
      "anchloc,dataloc,fcsloc,linkloc,listloc,mixedloc,nameloc,nmlist,
       nmquery,nmsploc,pathloc,proploc,queryloc,relloc,treeloc"
   >
]]>
<!entity %
   treetpel       -- Tree type associated element forms --
                  -- Clause: 7.5 --

   "pathloc,relloc,treeloc"
>
<!attlist
-- treetype --    -- Tree type --
                  -- Clause: 7.5 --
   (%treetpel;)

   treetype       -- Tree type --
                  -- Type of tree to address as location source --
      (content|subnode)
      content
>
]]><!-- treetype -->
```

7.6 Span Location Address

The attribute form **span location address** (*spanloc*) allows a multiple location address element to address a “span”. A span is a sequence of nodes selected from the list of nodes that occur contiguously between two locations (“span limits”) in the same grove in a left-list, pre-order traversal of the tree. A node is selected from this list (and is therefore a member of the span) if all of its content (or subnodes) are also in the list and it is not itself a descendant of a node in the span.

NOTE 161 For example, given the following tree:

```
        .---A---.
       /    |    \
      B     C     D
     / \    |    / \
    E  F   G   H  I
```

If the span start is node “F” and the span end is node “H”, the span addresses the nodes F, C, and H. The nodes A, B, and D are not included in the span because not all of their descendant nodes are included in the span. The node G is not included because it is a descendant of C.

The span limits are known as the “span start” and the “span end”. The span begins at the start of the span start and ends at the end of the span end, which must be no farther from the end of the document than the span start is. The span start and span end must both occur in the same tree rooted at a node in the location source. The span includes all nodes in that tree that occur after the span start and before the span end (inclusive) in a left-list, pre-order traversal of the tree.

If the location source is a node list, the location source for a particular span will be the node from the location source that is the root of the smallest tree that contains the span. The tree from which the span is selected is either a content tree or a subnode tree depending on the value of the tree type (treetype) attribute (see *7.5 Tree type*).

The attribute **span location** (*spanloc*) can specify that a multiple location address element is not a span location (“nspan”), or that it is a span location (“spanloc”); that is, that the element locates pairs of span limits.

When a location address is a span location address, it must address an even number of nodes, each pair being interpreted as a span start and span end.

It is an RHE to specify “spanloc” when the locations are not valid span limits.

The attribute **span limits sort** (*limsort*) specifies whether span limits are sorted so that the span start precedes the span end. The attribute is ignored if the spanloc attribute is specified as “nspan”.

NOTE 162 This attribute might allow a HyTime engine to optimize access to the span.

```
                        <!-- Span Location -->
<![ %spanloc; [
<!attlist
-- spanloc --     -- Span location attributes --
                  -- Clause: 7.6 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,mixedloc,nameloc,nmlist,
    nmquery,nmsploc,pathloc,proploc,queryloc,relloc,treeloc)

   spanloc        -- Span location --
                  -- Do multiple locations comprise a span? --
      (nspan|spanloc)
      nspan

   limsort        -- Span limits sort --
                  -- Are limits sorted with span start first? --
      (limsort|nlimsort)
      nlimsort
>
]]><!-- spanloc -->
```

7.7 Reference control

The attribute forms **reference type** (*reftype*) and **reference resolution control** (*refctl*) allow an application to control the target and resolution range of references. (Direct ID references are controlled by the ireftype attribute defined in the General Architecture of the SGML Extended Facilities; see *A.5.5 ID immediate referent type control*.)

A reference is an attribute whose declared value prescription is IDREF, IDREFS, ENTITY, or ENTITIES; an attribute or data content that is declared to contain references by a lextype attribute whose lexmodel includes IDREF, IDREFS, ENTITY, or ENTITIES; or an attribute or content defined as referential through the optional refloc facility (see *7.8 Reference location address*). Attribute ID references are specified by the attribute names, and content ID references by the reserved name “#CONTENT”. The reserved name “#ALL” means “all ID references in the attribute values and data content of this element type”.

Failure of references to conform to the reference resolution control attributes (refrange, reftype, etc.), or to conform in number or type to other attributes, are RHEs only when (and if) the application requires the references to be resolved sufficiently to recognize the failure. Such failures are always RHEs, however, if they can be determined from the markup alone (for example, if the number of IDREFS specified differs from the number prescribed by a reftype).

7.7.1 Reference element type

The attribute **reference element type** (*reftype*) associates names of referential attributes (or #CONTENT) with element types to which their ultimate targets must conform. The element types can be specified as a single GI, possibly followed by an occurrence indicator; a model group of GIs; or the reserved word "#ANY", which signifies that any target is acceptable for the corresponding attribute. If the refmodel option is supported, model groups can be any valid SGML model group. When the list of element types is not a repeating OR group or when it is a single GI without an occurrence indicator, the reference is limited to a single instance of one of the named types. If the refmodel option is not supported, model groups are limited to single (possibly repeatable) GIs or repeating or non-repeating OR groups of GIs.

NOTE 163 In other words, a simple list of possible target GIs.

The constraints associated with "#ALL" can be overridden for particular referential attributes by specifying element types for them individually.

The constraints are those of the target of the reference; intermediate location addresses can be used if permitted by the refrange attribute. If the constraint is a single GI, the element addressed must be of that element type. If the constraint is a model group, the elements addressed must satisfy the model group in the order they occur in the node list resulting from the address to which the reference type constraint applies.

NOTE 164 For example, if the reftype specification is "#ALL (B,C)" the reftype is satisfied by two elements, the first of whose element type is "B" and the second whose element type is "C". This node list could be addressed by two ID references, the first to element B, the second to element C, or by some indirect address that ultimately addresses the elements B and C in that order.

NOTE 165 The reftype attribute allows attribute values to be structures. The structures are defined as SGML element types, and instances of the structures are represented as elements. The structures do not occur directly in the attribute values; instead, references are used to point to them. The application designer can use the refrange attribute to require that the structures are referenced without indirect location addresses, and/or that they precede the reference in the document, in order to keep them close to the start-tags and to allow efficient processing.

NOTE 166 The reftype conventional comment performs the function of a "meta" reftype (with somewhat different syntax) for references defined by the HyTime language, as it constrains the architectural forms of ID reference targets, not their element types. An application designer can impose further constraints by specifying reftype attributes that will restrict the target element types.

```
                  <!-- Reference Element Type -->
<![ %reftype; [
<!attlist
-- reftype --     -- Reference type attributes --
                  -- Clause: 7.7.1 --
   #ALL

   reftype        -- Reference element type --
      CDATA       -- Lextype: (("#ALL",(GI|modelgroup|"#ANY"))?,
                               (ATTORCON,(GI|modelgroup|"#ANY"))*) --
                  -- Constraint: a given ATTNAME or #CONTENT can occur
                     only once; types apply to ultimate object of
                     address, not including any intermediate location
                     Address elements. --
                  -- Constraint: model groups limited to repeating
                     or non-repeating OR groups if refmodel option not
                     supported. Tokens in model groups must be GIs. --
      "#ALL #ANY" -- Constant --
>
]]><!-- reftype -->
```

7.7.2 Reference resolution range

The attribute **reference resolution range** (*refrange*) associates the names of referential attributes (or content) with resolution ranges. The resolution range associated with "#ALL" can be overridden for individual references by specifying their names with different ranges.

There are four resolution ranges, listed in order of increasing freedom; that is, the later ranges in the list also permit resolutions allowed by the ranges that precede them:

B **Backward direct reference**: An identified local element that precedes the reference in the same document (or subdocument).

D **Direct reference**: An identified local element in the same document (or subdocument) as the reference.

NOTE 167 A direct reference can be either backward or forward.

NOTE 168 Direct references can be used to address location address elements as though they were not location address elements. In other words, a location address element can be "mentioned" by a reference from an attribute whose reference range is set to "D".

I **Indirect reference**: A location in the same document (or subdocument) as the reference that is referenced indirectly through a location address element. This resolution range requires support of the location address module. If that module is not supported, this resolution range is treated like the "D" range, rather than being treated as an RHE.

X **External reference**: A location in any document (or subdocument). This resolution range requires support of the "exrefs" option. If that option is not supported, this resolution range is treated like the "I" range, rather than being treated as an RHE.

NOTE 169 The refrange facility allows the processing of elements that use referential attributes (for instance, hyperlinks) to be optimized. In other words, purely local references can be optimized.

7.7.3 Reference resolution level

The attribute **reference resolution level** (*reflevel*) associates the names of referential attributes (or content) with the maximum number of location path steps allowed for location paths referenced from those attributes (or content). The length of a branching location path is the length of the longest branch in the path. Counting begins with the first step referenced.

NOTE 170 In other words, the reference itself is not counted.

NOTE 171 The indirect reference pointers, from the initial reference to the target locations, comprise a "location path" that is normally transparent to the applications that process the target objects. The transparency of the indirect referencing allows the location path to be revised when an object is moved, without affecting the application. When a reflevel is specified, however, the steps of the location path are addressed, and this transparency may be lost.

The resolution level associated with "#ALL" can be overridden for individual references by specifying their names with different levels. A reference for which no maximum resolution level is specified (either by name or via "#ALL") is fully resolved.

A resolution level is specified in the form of an unsigned non-zero integer that represents the maximum number of location path steps relative to this element. If the target of a reference is itself a reference with a reflevel specified (or its target is, and so on), resolution is terminated at the earliest point specified by any of them.

NOTE 172 Some examples:

— If an element A has an IDREF with a reflevel of 6, resolution will cease 6 steps beyond it (7th level overall).

— If element B is a target of A's reference and has an IDREF with a reflevel of 3, resolution will cease 3 steps beyond B (4 steps beyond A, 5th level overall).

— If element C is a target of B's reference and has an IDREF for which no reflevel is specified (unlimited), resolution will cease 2 steps beyond C (4 steps beyond A, 5th level overall) because the earliest termination point specified (that of element B's IDREF) governs.

```
               <!-- Reference Resolution Control -->
<![ %refctl; [
<!attlist
-- refctl --      -- Reference resolution control attributes --
                  -- Clause: 7.7.3 --
   #ALL

   refrange       -- Reference resolution range --
                  -- Clause: 7.7.2 --
                  -- B  Backward reference to identified local element
                     D  Identified local element, before or after
                     I  Location address allowed
                     X  Indirect target, may be external to reference
                     --
      CDATA       -- Lextype: (("#ALL",("B"|"D"|"I"|"X"))?,
                              (ATTORCON,("B"|"D"|"I"|"X"))*) --
                  -- Constraint: a given ATTNAME or #CONTENT
                     can occur only once. Its declared value
                     or lextype must contain IDREF. --
      "#ALL X"    -- Constant --

   reflevel       -- Reference resolution level --
                  -- Level that IDREF resolution cannot exceed,
                     relative to this element. --
      CDATA       -- Lextype: (("#ALL",unzi)?,(ATTORCON,unzi)*) --
                  -- Constraint: a given ATTNAME or #CONTENT
                     can occur only once. Its declared value
                     or lextype must contain IDREF. --
      #IMPLIED    -- Default: resolve fully --
>
]]><!-- refctl -->
```

7.8 Reference location address

The attribute form **reference location address** (*refloc*) allows attributes whose semantics are addressing (that is, any attribute that would normally be declared with a value prescription of IDREF, IDREFS, ENTITY, or ENTITIES) to use any location address specification. The location address specification is interpreted as though the attribute were an ID reference to the equivalent location address element. The refloc attribute form is provided for element forms and notation forms.

The attribute **reference location type** (*loctype*) associates referential attributes with their location addressing methods. The value of the loctype attribute is a list of ATTORCON and keyword pairs. Each keyword defines the type of location address each attribute (or content) is using. The keywords are:

IDLOC the attribute value is interpreted as a name-space location in the location source document's element unique identifier name-space.

TREELOC the attribute value is interpreted as the content of a treeloc whose default location source is the referrer. Requires support for the treeloc option.

PATHLOC the attribute value is interpreted as the content of a pathloc whose default location source is the referrer. Requires support for the pathloc option.

RELLOC the attribute value is interpreted as a relative location address specified according to the normalized lexical type ((“ANC”|“CHILDREN”|“ESIB”|“FOLLOWNG”|“PARENT”|“PRECEDNG”|“ YSIB”),dimlist?), where the initial keyword specifies the relationship to the starting node, and the optional dimension list selects nodes from the list of nodes addressed by the specified relation. A dimension list may not be specified when the relationship is “parent” (see *7.10.1.6 Relative location address*). The referrer is the starting node. The principal tree root is the location source if no reference location source is specified. Requires support for the relloc option.

QUERYLOC This keyword must be followed by a notation name; the attribute value is interpreted as a query in the specified notation. Requires support for the queryloc option.

The value prescription of the attributes listed in the loctype value must be consistent with the location address specification assigned to them (CDATA always meets this requirement).

The value of a loctype attribute must be constant in a client document.

HyTime distinguishes attributes with a value prescription of IDREFS from attributes with a loctype specification of “IDLOC”. If an attribute has a declared value of IDREFS but is not also defined as an IDLOC location type, the attribute is referential, with each reference processed independently (as though each ID in the attribute value had been specified as a separate attribute). Attributes so declared do not require the multloc option. Attributes declared as IDREFS and also defined as an IDLOC are processed as a multiple location address such that all the nodes addressed by the ID references are treated as a single list of addressed nodes.

NOTE 173 In other words, for IDREFS that are not also IDLOCs, each ID reference is the beginning of an independent location path, while IDREFS that are also IDLOCs are taken as the beginning of a single, possibly multibranched location path. In grove terms, an IDREFS attribute that is not also an IDLOC addresses a list of node lists, one node list for each ID in the attribute value, while an IDREFS attribute that is also an IDLOC addresses a single node list representing the union of all the nodes directly or indirectly addressed by IDs in the attribute value.

Attributes with a value prescription of “ENTITY” or “ENTITIES” are always referential. Attributes with a value prescription of “ENTITIES” address a single node list whose members are the grove roots or principal tree roots of the groves constructed from the entities named in the attribute value.

NOTE 174 ENTITIES does not require the IDLOC distinction of IDREFS because entity references are always direct references to grove roots or principal tree roots and can therefore only result in a single node.

By default, the location source for referential attributes is either the document that contains the reference (for the IDLOC location type) or the element that makes the reference. A referential attribute can also be associated with another referential attribute that addresses the location source for the first attribute. The attribute **reference location source** (*rflocsrc*) associates a reference location attribute with another attribute of the same element that, when specified, addresses the location source for the referential attribute. The value of the rflocsrc attribute is a list of ATTORCON pairs, where the first name in the pair is a referential attribute or content and the second name is the referential attribute or content that addresses the location source for the first attribute or content (“#CONTENT” may not be specified for both members of a pair). If an attribute or content named as a reference location source is not specified for an element, the element itself is taken as the location source.

NOTE 175 Content is taken to not be specified only when the element has a declared content of EMPTY or exhibits an attribute with a default value prescription of #CONREF.

The value of the rflocsrc attribute must be constant in a client document.

It is not an error for one reference location source attribute to name another attribute of the same element as its reference location source. (In other words, it is possible to create a multi-rung location ladder with the referential attributes of a single element type.)

NOTE 176 Given a cross-reference element that uses two attributes, “target” and “doc”, to address elements in other documents, where target takes an ID reference and doc takes a reference to the document entity that contains the element with the ID named by the target attribute, the relationship between the target and doc attributes can be defined by the following loctype and rflocsrc attributes:

```
<!ATTLIST CrossRef
   HyTime   NAME     #FIXED "clink"
   HyNames  NAME     #FIXED "linkend target"

   -- Tell HyTime engine target is referential and uses ID references,
      either in this document or in another document. --
   loctype  CDATA    #FIXED "target IDLOC"
   rflocsrc CDATA    #FIXED "target doc"

   target   IDREFS   #REQUIRED

   -- DOC addresses location source of TARGET when specified. --
   doc      ENTITY   #IMPLIED
>
```

The attribute **span reference location** (*rflocspn*) names pairs of referential attributes that address the starts and ends of spans (see *7.6 Span Location Address*). The value of the rflocspn attribute is a list of ATTORCON pairs, where both names are referential attributes or content. The same attribute name (or “#CONTENT”) cannot be used for both members of a pair. When both attributes or content in the pair are specified, they are taken together to define a span. If only one member of the pair is specified, the address is interpreted as though the rflocspn attribute had not been specified. The value of the rflocspn attribute must be constant in a client document. The rflocspn attribute requires support for the spanloc facility, in addition to support for the refloc facility.

```
                 <!-- Reference Location Address -->
<![ %refloc; [
<!entity % rflocatt '
-- refloc --      -- Reference Location Address --
                  -- Clause: 7.8 --
   #ALL

   loctype        -- Reference location addresses type --
                  -- Each named attribute treated as if it were an
                     IDREF to a location address element. --
                  -- Constraint: The declared values of named
                     attributes must be lexically compatible with
                     their specified interpretation. --
                  -- Note: The declared value CDATA always meets this
                     requirement. --
      CDATA       -- Lextype: (ATTORCON,("IDLOC"|"TREELOC"|
                                         "PATHLOC"|"RELLOC"|
                                         ("QUERYLOC",NOTATION))+ --
      #IMPLIED    -- Constant --
                  -- Default: all references use SGML IDREFs, and each
                     IDREF in an IDREFS attribute is considered
                     separately --

   rflocsrc       -- Reference location source --
```

```
                  -- Associates referential attributes with their
                     location sources. --
      CDATA       -- Lextype: (ATTORCON,ATTORCON)+ --
                  -- Constraint: attributes named must be referential
                     attributes. --
      #IMPLIED    -- Constant --
                  -- Default: all referential attributes have this
                     element as their location source. --
'>
<!attlist %rflocatt; >
<!attlist #NOTATION %rflocatt; >
]]><!-- refloc -->


                   <!-- Reference Location Span -->
<![ %refloc; %spanloc; [
<!entity % rfspnatt '
-- rflocspn --    -- Reference location span --
                  -- Clause: 7.8 --
   #ALL

   rflocspn       -- Reference location span --
                  -- Names pairs of referential attributes that
                     address spans when both attributes are
                     specified. --
      CDATA       -- Lextype: (ATTORCON,ATTORCON)+ --
                  -- Constraint: attributes named must be referential
                     attributes. --
      #IMPLIED    -- Constant --
'>
<!attlist %rfspnatt; >
<!attlist #NOTATION %rfspnatt; >
>
]]><!-- refloc, spanloc -->
```

NOTE 177 The refloc facility has a number of useful applications, including:

— Fixing the values of ID references in document types.

— Using non-HyTime location address notations directly; for example, using URLs with HyTime hyperlinks:

```
          <!-- HTML A (anchor) element (derived from HTML 3.2) -->
   <!NOTATION URL
      PUBLIC "-//IETF/RFC1738//NOTATION Uniform Resource Locator//EN"
   >
   <!ELEMENT A
      - O (%text;)* -(A)
   >
   <!ATTLIST A
      HyTime   NAME     #FIXED clink
      HyNames  CDATA    #FIXED "linkend href"
      anchcstr NAMES    #FIXED "self cond"
      linktrav NAMES    #FIXED "A D"
      loctype  CDATA    #FIXED "href QUERYLOC URL"

      name     CDATA    #IMPLIED -- named link end --
      href     %URL     #IMPLIED -- URL for linked resource --
      rel      CDATA    #IMPLIED -- forward link types --
      rev      CDATA    #IMPLIED -- reverse link types --
      title    CDATA    #IMPLIED -- advisory title string --
   >
```

— Retrofitting HyTime to preexisting addressing schemes such as the common target/doc example shown in note 176.

7.9 Name-space locations

A name-space location addresses a node by a unique name in some name-space. For SGML documents, the primary name-spaces are elements with IDs (represented by the elements property of the SGMLDOC node) and entities (represented by the entities property of the SGMLDOC node). In grove terms, a name-space is a property whose value is a named node list. A named node list is a node list in which all the nodes exhibit a name property, for which each node in the named node list must exhibit a unique value. Thus, the location source for a name-space location address is implicitly a property location address that addresses a property whose value is a named node list.

7.9.1 Identified local element or entity

A reference can be made directly to an element in the same document that has an ID attribute simply by specifying the ID. A location address element is not needed in such a case. A reference can be made directly to an entity declared in the same document by naming the entity.

7.9.2 Property location address

The element form **property location address** (*proploc*) addresses the value of the property of a node in a grove by property name. The location source of a property location address is the node or nodes that exhibit the property whose value is addressed. The property is named in the content of the proploc element.

The attribute **additional property source** (*apropsrc*) specifies whether the processor should examine the origin ancestry of the location source (starting with the location source's origin) for the property named in the content of the property location if it is not exhibited by the location source (apropsrc) or whether the location source is the only source for the property (solesrc).

NOTE 178 For example, if the property is “elements”, the location source is the document element of an SGML document, and the additional property source value is “apropsrc”, the HyTime engine will use the elements property exhibited by the SGMLDOC node, which is the origin of the document element node.

The attribute **not an applicable property** (*notprop*) specifies the treatment of a proploc when the property named is not exhibited by the location source or (if specified) its additional property source. The proploc can be treated as an RHE (“error”) or it can be ignored (“ignore”).

NOTE 179 The SGML property set is defined in *A.7 SGML Property Set*.

The attribute **direct or indirect value of property** (*direct*) specifies whether the value of the property should be the direct value (the value in the SGML document grove) or the indirect value (the value addressed by value reference), if there is one (see *6.7.1 Value Reference*). The default is to return the indirect property value if there is one. The attribute is ignored if the property does not have an indirect value.

```
                <!-- Property Location Attributes -->
<![ %proplat; [
<!attlist
-- proplat --     -- Property location attributes --
                  -- Clause: 7.9.2 --
   (nmsploc,proploc)

   apropsrc       -- Use additional property source? --
```

```
      (apropsrc|solesrc)
      solesrc

   notprop        -- If not a property of locsrc or apropsrc? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   direct         -- Direct or indirect value --
                  -- Constraint: Ignored if no indirect value --
      (direct|indirect)
      indirect
>
]]><!-- proplat -->

                  <!-- Property Location Address -->
<![ %proploc; [
<!element
   proploc        -- Property location address --
                  -- Clause: 7.9.2 --
                  -- Locates property of a grove node --
   - O
   (#PCDATA)      -- Lextype: compname --

-- Attributes [locs]: proplat, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % proplat "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- proploc -->
```

7.9.3 Name-space location address

The element form **name-space location address** (*nmsploc*) addresses nodes in named node lists. The location source for a name-space location address must be a node that exhibits (or whose additional property source exhibits) a named node list property. The name-space location address selects nodes from the value of the named node list property.

NOTE 180 In other words, a name-space location address includes an implicit property location that addresses the value of the specified named node list property.

Because it performs an implicit property location, the name-space location address form uses the proploc attributes (see *7.9.2 Property location address*).

The content of nmsploc is a s-separator-delimited list of zero or more names. Names that include separators may be specified as parameter literals.

There is no resolution of a name, nor checking of its validity as an ID or entity name, until the name-space location address is referenced in a manner that requires access to the object addressed by the name.

The attribute **name-space property** (*namespc*) names the name-space property of the location source from which nodes will be selected by name.

The attribute **additional property source** (*apropsrc*) (from the proploc attribute form) specifies whether or not the processor should examine the origin ancestry of the location source (starting with the location source's origin) for the property named by the namespc attribute if it is not exhibited by the location source (apropsrc).

NOTE 181 For example, if the name-space property is “elements”, the location source is the document element of an SGML document, and the additional property source value is “apropsrc”, the HyTime engine will use the elements property exhibited by the SGMLDOC node, which is the origin of the document element node.

The attribute **not a name-space** (*notspc*) indicates the action to take when the property named is not a name-space property: return an error condition (error) or ignore the location address (ignore).

The attribute **not a name** (*notname*) indicates the action to take when when a name does not exist in the location source: return an error condition (error) or ignore the location address (ignore).

A name-space location address with an empty list of names addresses an empty node list.

When the location source is a node list, the name-space location address is applied to each node in the list, creating a single list of nodes.

```
                 <!-- Name-Space Location Address -->
<![ %nmsploc; [
<!element
   nmsploc        -- Name-space location address --
                  -- Clause: 7.9.3 --
                  -- Addresses objects by name in a name-space --
                  -- Constraint: location source must be an object that
                     exhibits a named node list property or whose
                     additional property source exhibits a named node
                     list property. --
   - O
   (#PCDATA)      -- Lextype: (word|literal)* --

-- Attributes [locs]: nmsploc, proplat, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nmsploc        -- Name-space location address --
                  -- Clause: 7.9.3 --

   namespc        -- Name-space --
                  -- Name-space property of location source from which
                     nodes are selected. --
      NAME
      #REQUIRED

   notspace       -- If nmspace name is invalid? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   notname        -- If name is not valid in nmspace? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % proplat "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- nmsploc -->
```

7.9.4 Mixed location address

The element form **mixed location address** (*mixedloc*) addresses objects by containing other location address elements. Thus a mixedloc is equivalent to putting an ID on each contained location address element and creating a name-space location address whose name list lists those IDs (and only those IDs) and whose location source is the SGMLDOC node of the grove for the document in which the mixedloc occurs.

```
                   <!-- Mixed Location Address -->
<![ %mixedloc; [
<!element
   mixedloc       -- Mixed location address --
                  -- Clause: 7.9.4 --
                  -- Groups location addresses together. --
   - O
   (%loc;)*

-- OptionalAttributes [locs]: multloc, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
]]><!-- mixedloc -->
```

7.9.5 Named location address

The element form **named location address** (*nameloc*) is a specialized mixed location address that contains only name list or nmquery elements. The name list element form is derived from the name-space location address element form (see *7.9.3 Name-space location address*). The nmquery element form is derived from the query location address element form (see *7.11.1 Query location address*).

```
                   <!-- Named Location Address -->
<![ %nameloc; [
<!element
   nameloc        -- Named Location Address --
                  -- Clause: 7.9.5 --
                  -- Assigns a local ID to one or more named objects --
   - O
   (nmlist|nmquery)*

-- Attributes [locs]: nameloc --
-- OptionalAttributes [locs]: multloc, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nameloc        -- Named Location Address --
```

```
                  -- Clause: 7.9.5 --

   HyBase   NAME     #FIXED mixedloc
>
]]><!-- nameloc -->
```

7.9.6 Name list specification

The element form **name list specification** (*nmlist*) contains an undelimited SGML attribute value specification for a name list. The nmlist form is derived from the nmsploc form.

There is no limit on the number of names. The names are derived from the name list attribute value specification in the manner defined by ISO 8879.

NOTE 182 Leading and trailing separators are ignored, and intervening ones are reduced to a single space, resulting in a space-delimited name list.

The attribute **name type** (*nametype*) indicates whether the names in the list are entity names or element IDs. (The nametype attribute is a synonym for the namespc attribute of the nmsploc element form.)

If the name specification list is empty, it is considered to address the SGMLDOC node of either the document named by the docorsub attribute, if specified, or the document in which the nmlist occurs.

NOTE 183 Support of the "queryloc" option is *not* required for this purpose.

The attribute **SGML document or subdocument** (*docorsub*) identifies the SGML document entity of the SGML document, or the SGML subdocument entity of the SGML subdocument, in which the names in the name list are defined. Unless the "anysgml" option is supported, the specified document or subdocument, and the document in which this element occurs, must be subject to equivalent SGML declarations. (The docorsub attribute is a synonym for the locsrc attribute from the location source attribute list.)

```
                   <!-- Name List Specification -->
<![ %nameloc; [
<!element
   nmlist         -- Name list specification --
                  -- Clause: 7.9.6 --
                  -- Addresses elements or entities in an SGML document --
   - O
   (#PCDATA)      -- Reference --
                  -- Lextype: (IDREF|ENTITY)* --

-- Attributes [locs]: nmlist, proplat, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nmlist         -- Name list specification --
                  -- Clause: 7.9.6 --

   HyBase   NAME     #FIXED nmsploc
   impsrc   NAME     #FIXED grovert
   HyBnames CDATA    #FIXED "locsrc docorsub
                             namespc nametype
```

```
                                #MAPTOKEN elements element
                                #MAPTOKEN entities entity"

   nametype       -- Name-space from which nodes are selected --
      (entity|element)
      entity

   docorsub       -- Document or subdocument location source --
      ENTITY
      #IMPLIED

   notspace       -- If nmspace name is invalid? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   notname        -- If name is not valid in nmspace? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % proplat "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- nameloc -->
```

7.10 Coordinate locations

Coordinate locations are element forms that address sequences of nodes in lists or trees by a corresponding number of dimension specifications.

See *6.8 Coordinate Specifications* for more information about about coordinate and dimension specifications.

7.10.1 Node locations

Node locations are coordinate locations in which the quantum is a node in a node list or tree. A node has the ability to be viewed both as a member of a node list and as the root of a tree (either a subnode tree or, if the node has a children property, the root of a content tree). This ability gives rise to four types of node location address:

— A “list location address” (listloc) addresses one or more nodes selected from a node list. Its location source is a node list.

— A “tree location address” (treeloc) addresses a single node of a tree in the classical manner by selecting an ancestor node at each level. Its location source is a node list, each node in the list viewed as the root of a tree.

— A “path location address” (pathloc) addresses a node list comprising one or more nodes selected from a tree, by viewing the tree as a matrix. Its location source is a node list, each node in the list viewed as the root of a tree.

— A “relative location address” (relloc) addresses a node list comprising one or more nodes selected from a tree by virtue of their relationship to a given starting node. Its location source is a node list, each node in the list viewed as the root of a tree.

7.10.1.1 Node lists

The fundamental organizational structure for nodes in groves is a node list. HyTime addresses nodes in a node list by treating the node list as a single-axis coordinate space where the nodes are the quanta. Sequences of nodes are addressed by a corresponding number of dimension specifications. The result of resolving any location address is always a node list (possibly empty).

NOTE 184 The largest node list that can be addressed is determined by the value of the highest quantum count limit specified by the HyTime architecture use declaration (see *6.3 HyTime support declarations*).

7.10.1.2 List location address

The element form **list location address** (*listloc*) is a coordinate location address that addresses nodes selected from a node list provided by its location source. Its location source is a node list, viewed as a list.

```
                    <!-- List Location Address -->
<![ %listloc; [
<!element
   listloc        -- List location address --
                  -- Clause: 7.10.1.2 --
                  -- Locates nodes in a node list --
   - O
   (%dimlist;)*

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- listloc -->
```

7.10.1.3 Tree combination

When the location source for a tree or path address is a node list, each node in the list is normally taken as the root of an independent tree, with the location address applied to each tree in turn. The result is a single node list consisting of the nodes selected from each tree in the order the tree roots occurred in the location source node list. However, the attribute form *treecom* can be used with a multiple location source to view the node list as members of a single tree with a common (imaginary) parent node, thereby combining the multiple source trees into one. The results are then the same as for a non-multiple source: a single node in the case of treeloc, and a node list in the case of pathloc. This allows tree-based location addresses to select a single tree from a multiple location source.

NOTE 185 As the root level is included in the tree addressing, a particular root node can be selected when multiple source trees are combined.

The attribute **tree combination** (*treecom*) specifies whether multiple source trees are combined (“treecom”) or not (“ntreecom”).

```
                 <!-- Tree Combination Attributes -->
<![ %treecom; [
<!attlist
-- treecom --     -- Tree combination attributes --
                  -- Clause: 7.10.1.3 --
   (pathloc,treeloc)
   treecom        -- Tree combination --
                  -- Combine multiple source trees --
      (ntreecom|treecom)
      ntreecom
>
]]><!-- treecom -->
```

7.10.1.4 Tree location address

The element form **tree location address** (*treeloc*) addresses a single node of a tree in the classical manner by selecting a node from an addressable range at each level, starting at the root. The tree addressed is either a content tree or subnode tree, depending on the value of the treetype attribute (see *7.5 Tree type*).

The nodes are selected by single axis markers applied to successive addressable ranges. Each marker acts as the first marker of a dimension specification; the second marker is not specified and is an implied “1”. Therefore, each marker selects a single node from the addressable range.

The addressable range of the root level (unless there are multiple roots) is a single node. The addressable range of the second level is the children (or subnodes) of the root node. It is addressed by a single axis marker that locates a node within it. Lower levels of the tree are addressed by the recursive application of successive axis markers to the children (or subnodes) of the last selected node.

The list of axis markers used to address nodes in a tree is a “tree position specification” (treepos).

NOTE 186 For example, in a document with the following structure:

```
<!--                     tree position -->
<book>                   <!-- 1        -->
  <body>                 <!-- 1 1      -->
    <chapter>            <!-- 1 1 1    -->
      <para></para>      <!-- 1 1 1 1  -->
      <para></para>      <!-- 1 1 1 2  -->
    </chapter>
  </body>
  <annex>                <!-- 1 2      -->
    <figure></figure>    <!-- 1 2 1    -->
    <figure></figure>    <!-- 1 2 2    -->
    <figure></figure>    <!-- 1 2 3    -->
  </annex>
</book>
```

the third “figure” element can be identified with respect to the “book” as the root, by the following tree location specification:

```
<treeloc>1 2 3</treeloc>
```

where “1” selects the book (the root), “2” selects the annex (the second child of the root), and “3” selects the third figure (the third child of the annex).

```
                    <!-- Tree Location Address -->
<![ %treeloc; [
<!element
   treeloc        -- Tree location address --
                  -- Clause: 7.10.1.4 --
                  -- Locates nodes in a tree by classical method --
   - O
   (%marklist;)*  -- Constraint: resolved axis markers are interpreted
                     as a single list of dimension specifications,
                     marker1 only; implied marker2 is 1. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treecom,
   treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- treeloc -->
```

7.10.1.5 Path location address

The element form **path location address** (*pathloc*) is a coordinate location address that addresses a tree as though it were a matrix in which the matrix rows are the levels of the tree and the matrix columns are the paths. Path location address requires support of the “pathloc” option and “multloc” option in addition to the options and modules required for all coordinate locations.

In HyTime, a path of a tree is the sequence of nodes comprising the ancestry of a given leaf, including the root node from which the leaf is descended. The tree is viewed as a “path list” consisting of all of the paths, ordered in the left-list pre-order of their leaves.

NOTE 187 Each leaf appears in exactly one path. A node that is not a leaf appears in as many paths as there are leaves descended from it.

The path list is addressed by pairs of dimension specifications, contained in the dimension specification list.

— The first dimension specification of the pair selects paths. Its addressable range is the path list.

— The second dimension specification selects nodes within each of the selected paths concurrently. Each path is a separate addressable range whose first quantum is a root node and whose last quantum is a leaf. The dimension specification must must avoid overrunning the shortest path unless “trunc” is specified for the overrun attribute of the pathloc.

NOTE 188 In terms of the analogy to a matrix, the first dimspec of a pair selects one or more consecutive columns (paths to leaves), while the second selects one or more rows in the selected columns.

For example, to select:

— All leaves.

```
<dimspec>1 -1</dimspec>
<dimspec>-1 1</dimspec>
```

— All children and grandchildren of the roots.

```
<dimspec>1 -1</dimspec>
<dimspec>2 2</dimspec>
```

— All grandchildren of the first child (that is, the earliest child of the first root in a pre-order traversal), assuming that exactly 3 leaves descend from that child and that the leaves are its grandchildren or descendants of its grandchildren.

```
<dimspec>1 3</dimspec>
<dimspec>4 1</dimspec>
```

— All nodes except the roots and their children

```
   <dimspec>1 -1</dimspec>
   <dimspec>3 -1</dimspec>
```

NOTE 189 In the “book” example in *7.10.1.4 Tree location address*, the third figure could be located by specifying a pathloc for the last leaf of the tree, as follows:

```
<dimspec>-1 1</dimspec>   <!-- Last path -->
<dimspec>-1 1</dimspec>   <!-- Last leaf -->
```

The selected nodes are formed into a node list in the order they would be visited in a left-list pre-order traversal of the tree. If the multloc attribute “set” is specified, each node appears only once in the node list, even though it may have been selected from multiple paths in which it occurred.

NOTE 190 The dimension specification list can be produced by a marker function that locates nodes anywhere in the tree. For example, in an SGML grove, it could test generic identifiers to locate all nodes that are “chapter title” elements.

```
                    <!-- Path Location Address -->
<![ %pathloc; [
<!element
   pathloc         -- Path location address --
                   -- Clause: 7.10.1.5 --
                   -- Locates nodes in a tree viewed as a path list --
   - O
   (%dimlist;)*    -- Constraint: resolved axis markers are interpreted
                      as a single list of pairs of dimension
                      specifications. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treecom,
   treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- pathloc -->
```

7.10.1.6 Relative location address

The element form **relative location address** (*relloc*) is a coordinate location address that addresses nodes of a tree in terms of their relationship to another node.

The attribute **starting node** (*strtnode*) addresses the node whose relatives are to be addressed. The location source addresses the root of the tree in which the starting node occurs. It is an RHE if the starting node does not occur in the tree rooted at the location source. If the location source is a node list with more than one node, each node is taken to be the root of an independent tree. The attribute **no starting node** (*nostart*) indicates whether it should be considered an error if a tree is addressed as a location source for which no starting node is addressed (error) or whether the tree should be ignored (ignore). If the strtnode attribute is omitted, the nodes in the location source are used as the starting nodes.

The attribute **relationship to starting node** (*relation*) specifies an addressable range of the tree from which nodes are selected.

The content of the relloc is a dimension specification that selects nodes from the node list specified by the relation attribute. The following list states, for each allowable relation, the name and the addressable range it represents. The nodes in the addressable range and the nodes selected are always listed in the order of left-list pre-order traversal of the tree.

anc — Ancestor: the ancestral nodes from the location source to the parent of the starting node.

NOTE 191 Examples: (1 1)=root (-1 1)=parent

esib — Elder sibling: the elder (left side) siblings of the starting node

NOTE 192 Examples: (1 1)=oldest esib (-1 1)=youngest esib

ysib — Younger sibling: the younger (right side) siblings.

NOTE 193 Examples: (1 1)=oldest ysib (-1 1)=youngest ysib

parent — Equivalent to “anc” with a dimension specification of (-1 1). Actual dimension specification must be empty.

children — The children of the starting node.

NOTE 194 Examples: (1 1)=first child, (-1 1)=last child

precedng — All nodes in the tree that occur before the starting node.

followng — All nodes in the tree that occur after the starting node.

```
                  <!-- Relative Location Address -->
<![ %relloc; [
<!element
   relloc         -- Relative location address --
                  -- Clause: 7.10.1.6 --
                  -- Locates nodes in a tree relative to a starting
                     node --
   - O
   (%dimlist;)*   -- Constraint: resolved axis markers are interpreted
                     as a single list of dimension specifications. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc, relloc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   relloc         -- Relative location address --
                  -- Clause: 7.10.1.6 --

   strtnode       -- Starting node(s) --
                  -- Starting node(s) whose relatives are to be addressed --
      CDATA       -- Reference --
      #IMPLIED    -- Default: root(s) of location source tree(s) --

   nostart        -- No start --
```

```
                  -- Expected system behavior if no starting node is addressed
                     for location source. --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   relation       -- Relationship to starting node --
      (anc|children|esib|followng|parent|precedng|ysib)
      parent
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- relloc -->
```

7.10.2 Data location address

The element form **data location address** (*dataloc*) addresses character data that has been tokenized by a data tokenizer grove construction process using the lexical type name or HyLex match string specified by the *filter* attribute of the dataloc element form. The dataloc form uses a coordinate specification to select tokens from the full list of tokens derived from the location source. (See *A.4.4.2 Data tokenizer (DATATOK) grove construction* for a description of data tokenizer grove constructors and groves.)

NOTE 195 In other words, a dataloc is equivalent to applying a list location address to the list of tokens created by a data tokenizer grove construction process and then either returning the nodes containing the tokens or returning the list of data character nodes that are the source of the tokens. The complete location ladder to which dataloc is equivalent is shown below.

NOTE 196 The plain text grove construction process can be used to create a grove from unstructured character data to which a dataloc or data tokenizer grove construction process can be applied. This could be done either by creating an explicit grovedef element or by associating the plain text grove construction process with a data entity containing the text to be tokenized and then naming the entity as the location source of the data location address.

The data tokenizer notation for the dataloc form may be either the HyTime data tokenizer notation (HyDatTok) or the HyTime lexical tokenizer notation (HyLexTok). Both HyDatTok and HyLexTok are derived from the data tokenizer grove construction (datatok) notation form (see *A.4.4.2 Data tokenizer (DATATOK) grove construction*). When the data tokenizer notation is HyDatTok, only the lexical types defined below may be used as values for the filter attribute. When the data tokenizer notation is HyLexTok, any HyLex expression may be used as a filter value. The HyLexTok notation's notation derivation source (superdcn) is the HyTime lexical model (HyLex) notation (see *A.2.3 HyTime lexical model notation (HyLex)*). The datatok notation provides the semantics of grove construction for both HyDatTok and HyLexTok, while the HyLex notation provides the syntax for the match token specification for HyLexTok.

The attribute **tokenization filter** (*filter*) specifies the match expression the data tokenizer process will use to recognize tokens in the location source.

NOTE 197 A lexical type name, such as any of the values allowed for HyDatTok, is a valid HyLexTok match expression.

The following lexical types are defined in the HyTime lexical type definition document specifically for use with the dataloc element form and must be supported by systems that support the dataloc form but do not support lexical typing. In each of these types, the term “character” means any of the following node types: datachar, sdata, extdata, and subdoc. These node types are all treated as characters semantically.

str Unnormalized string: each character in the location source is a token.

norm Normalized text: all characters except RS, RE, and SEPCHAR (white space).

NOTE 198 Note that, in the reference concrete syntax, “Charles' and Linda's Abyssinians” is four normalized tokens.

word Word tokens: all name characters.

NOTE 199 Note that, in the reference concrete syntax, “Charles' and Linda's Abyssinians” is five word tokens.

NAME SGML name: NAME constructed according to ISO 8879.

sint Signed integer: digit string, optionally preceded by hyphen or plus.

UTCdate UTC date: Valid date in UTC yyyy-mm-dd format.

UTCtime UTC time: Valid time in UTC hh:mm:ss.decimal format.

UTC UTC date and time: pairs of DATE and TIME tokens, DATE first.

line Each token consists of the characters occurring between RS/RE pairs, between a tag close and a RE, or between an RS and a tag open.

The attribute **tokens or data** (*tokordat*) specifies whether the data location address returns the token string nodes in the data tokenizer grove (tokens) or the data characters in the location source corresponding to the tokens (data). The default is to return the data.

```
                  <!-- HyTime Lexical Tokenizer -->
<![ %HyLexTok; [
<!notation
   HyLexTok       -- HyTime Lexical Tokenizer --
                  -- A data tokenizer grove constructor using the
                     HyLex syntax. --
                  -- Clause: 7.10.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION HyTime Lexical Tokenizer//EN"

-- Attributes [locs]: HyLexTok --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyLexTok       -- HyTime Lexical Tokenizer --
                  -- Clause: 7.10.2 --

   HyBase    NAME     #FIXED datatok
   superdcn  NAME     #FIXED HyLex

-- Attributes from datatok notation --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED

-- Attributes from HyLex notation --
   norm     (norm|unorm) norm
>
<!attlist
-- dlhylex --     -- HyLex attributes for dataloc and datatok --
```

```
               -- Clause: 7.10.2 --
   (dataloc,datatok)

-- Attributes from HyLex notation --
   norm     (norm|unorm) norm
>
<!entity % HyLex "INCLUDE">
]]><!-- HyLexTok -->
]]>

               <!-- Data Tokenizer Grove Definition -->
<![ %datatok; [
<!notation
   HyDatTok       -- HyTime Data Tokenizer --
                  -- A simple data tokenizer grove constructor. --
                  -- Clause: 7.10.2 --
                  -- Lextype: ("LINE"|"NAME"|"NORM"|"SINT"|"STR"|
                               "UTC"|"UTCDATE"|"UTCTIME"|"WORD") --

   PUBLIC "ISO/IEC 10744:1997//NOTATION HyTime Data Tokenizer//EN"

-- Attributes [locs]: HyDatTok --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDatTok       -- HyTime Data Tokenizer --
                  -- Clause: 7.10.2 --

   HyBase    NAME     #FIXED datatok

-- Attributes from datatok form --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
<![ %HyLexTok; [
   <!entity % ddattok "HyLexTok">
]]>
<!entity %
   ddattok        -- Default data tokenizer notation --
                  -- Clause: 7.10.2 --

   "HyDatTok"
>
<!element
   datatok        -- Data tokenizer grove definition --
                  -- Clause: 7.10.2 --
                  -- Generates a datatok grove from data in another
                     grove. --
   - O
   (#PCDATA)
```

```
-- Attributes [locs]: datatok --
-- OptionalAttributes [locs]: dlhylex
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   datatok        -- Data location address --
                  -- Clause: 7.10.2 --

   HyBase   NAME     #FIXED agrovdef

-- Attributes from agrovdef --
   grovesrc CDATA    #REQUIRED
   grovecon NOTATION (HyDatTok|HyLexTok) #FIXED %ddattok;

-- Attributes from datatok notation form (via DAFE) --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
]]><!-- datatok -->

                    <!-- Data Location Address -->
<![ %dataloc; [
<!element
   dataloc        -- Data location address --
                  -- Clause: 7.10.2 --
                  -- Locates string and token data objects in data --
   - O
   (%dimlist;)*   -- Constraint: interpreted as a single list of
                     dimension specifications --

-- Attributes [base]: overrun --
-- Attributes [locs]: dataloc, locsrc, impsrc --
-- OptionalAttributes [locs]: dlhylex, multloc, referatt, spanloc,
   treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dataloc        -- Data location address --
                  -- Clause: 7.10.2 --

   HyBase   NAME     #FIXED datatok
   HyBnames CDATA    #FIXED "HyBase #DEFAULT
                             #ARCCONT filter
                             grovesrc locsrc"

   filter         -- Tokenization filter --
      CDATA       -- Constraint: If no data tokenizer notation
```

```
                      specified, filter must be a single lexical type
                      name. --
      "str"

   tokordat        -- Result to return: tokens or source data? --
      (data|tokens)
      data

-- Attributes from datatok form --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- dataloc -->
```

NOTE 200 Any data location address can be replaced by a location ladder that uses an explicit grove definition to associate a data tokenization grove construction process with source data to be tokenized, as shown below (see *A.4.4.2 Data tokenizer (DATATOK) grove construction*). The “datatok” element is an auxiliary grove definition (agrovdef) element that uses the dataloc's location source as the grove source for the grove constructor and the dataloc's filter attribute value for the datatok's tokenization filter (mapped to an attribute using the DAFE facility's NotNames attribute; see *A.5.3 Data Attributes for Elements (DAFE)*). The content and relevant location-addressing attributes of the dataloc become the content and attributes of the listloc that addresses the tokens created by the data tokenizer.

The location ladder shown below works as follows:

— The datatok element produces the data tokenizer grove by tokenizing the data content of the grove's source text (which corresponds to the location source of the dataloc).

— The first proploc element addresses all the tokenstr nodes in the data tokenizer grove, which make up the content of the strings property of the tokenized root (tokroot) node.

— The listloc element selects tokenized string (tokenstr) nodes from the list returned by the proploc. The coordinate address used by the listloc corresponds to the coordinate address specified by the dataloc. If the value of the dataloc's tokordat attribute is “tokens”, this is the list returned by the dataloc.

— The second proploc element addresses the source properties of the tokenstr nodes returned by the listloc element. The result is a list of data character nodes from the original source text nodes corresponding to the characters of the tokens selected by the listloc. If the value of the dataloc's tokordat attribute is “data”, this is the list returned by the dataloc.

```
<!DOCTYPE DatalocDef [

<?IS10744 ArcBase HyTime>
<!NOTATION HyTime
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!ATTLIST #NOTATION HyTime
   ArcDocF  NAME     #FIXED HyDoc
   ArcDataF NAME     #FIXED HyBrid
   ArcDTD   CDATA    #FIXED "HyTime"
   ArcQuant CDATA    #FIXED "NAMELEN 9"
   ArcOpt   CDATA    #FIXED "locs"
   locs     CDATA    #FIXED "proploc nmsploc pgrovdef"
>
<!NOTATION SGML
   PUBLIC "ISO 8879:1986//NOTATION Standard Generalized Markup Language//EN"
>
<!ENTITY HyTime
   PUBLIC "ISO/IEC 10744:1997//DTD AFDR Meta-DTD
```

```
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
   CDATA SGML
>

<!ENTITY %
   dtokatt        -- Data tokenizer attributes --
                  -- Derived from the data tokenizer grove
                     construction notation in the Property Set
                     Definition Requirements annex. --

   "catsrc   CDATA    #IMPLIED
    cattoken CDATA    #IMPLIED
    catres   CDATA    #IMPLIED
    boundary (sodeod|sodiec|isceod|isciec|inmodel) sodeod
    notation NOTATION (datatok) #FIXED datatok"
>

<!NOTATION datatok
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Datatok Grove Construction Process//EN"
>
<!ATTLIST #NOTATION datatok
   %dtokatt;
>

<!ELEMENT datalocdef
   O O (datatok,(proploc|listloc)*)
>

<!ELEMENT datatok
   - O EMPTY
>
<!ATTLIST datatok
   id       ID       #REQUIRED
   grovesrc IDREFS   #REQUIRED
   filter   CDATA    "str"

-- Data tokenizer attributes will be interpreted as data tokenizer
   grove construction attributes via DAFE facility rules. --
   %dtokatt;

   grovecon NOTATION (datatok) datatok
   HyTime   NAME     #FIXED agrovdef
   NotNames CDATA    #FIXED "#NOTCONT filter"
>

<!ELEMENT proploc - O  (#PCDATA)
>
<!ATTLIST proploc
   ID       ID       #REQUIRED
   locsrc   IDREF    #REQUIRED
   HyTime   NAME     #FIXED "proploc"
>
<!ELEMENT listloc - O  (#PCDATA)
>
<!ATTLIST listloc
   ID       ID       #REQUIRED
   locsrc   IDREF    #REQUIRED
   HyTime   NAME     #FIXED "listloc"
>

]>
<datatok
   id       = datatok-grove
   grovesrc = source-nodes -- Replace with location source of dataloc --
   filter   = word
>
<proploc
   locsrc   = datatok-grove
   id       = strings-property
   >STRINGS
</proploc>
<listloc
   locsrc   = strings-property
   id       = token-nodes
   >1 -1   <!-- Acts the same as dimspecs from dataloc -->
</listloc>
<proploc
   locsrc   = token-nodes
   id       = token-data
   >SOURCE
</proploc>
```

7.11 Querying

HyTime recommends the use of the Standard Document Query Language (SDQL), defined in the DSSSL standard, ISO/IEC 10179:1996 Document Style Semantics and Specification Language, for the queryloc and nmquery element forms. The SDQL language includes equivalents of all the HyTime location address forms.

NOTE 201 This means, in part, that any location ladder can be replaced by the equivalent SDQL query.

However, any query notation can be used with HyTime as long as the notation can operate on groves and returns node lists as the result of the query.

NOTE 202 Any existing query mechanism can be integrated into a HyTime system by defining an appropriate property set and grove construction process for the data notation it is designed to work with.

HyTime defines a general query-based location address, **queryloc**, that can be used both for general queries and as the base for specialized location addresses derived from the queryloc form. The hyperlinks module defines two queryloc-form location addreses, linkloc and anchloc, for addresing hyperlinks and their anchors according to their HyTime-specific semantics as reflected in the HyTime architectural semantic grove (see *8.3 Hyperlink-related location addresses*). The scheduling module defines a queryloc-form location address, fcsloc, for addressing events in event schedules (see *9.10 Finite coordinate space location address*).

7.11.1 Query location address

The element form **query location address** (*queryloc*) contains a query that returns a node list.

If the location source for a queryloc is a multiple location and is not a span location, each node in the location source is treated as the root of a separate query domain.

The attribute **query notation** (*notation*) specifies the data content notation governing the query specified by the query location address.

NOTE 203 The data attributes for elements (DAFE) facility can be used to provide notation-specific attributes for queryloc-form elements (see *A.5.3 Data Attributes for Elements (DAFE)*).

The attribute **no data found** (*notfound*) indicates the expected system response when the query fails to return any nodes: either report an RHE (“error”) or ignore the query location address (“ignore”).

```
                     <!-- Query Location Address -->
<![ %queryloc; [
<!element
   queryloc       -- Query location address --
                  -- Clause: 7.11.1 --
                  -- Locates objects in a grove using queries against
                     their properties as defined in the property set
                     used to construct the grove. --
   - O
   (%HyCFC;)*

-- Attributes [locs]: locsrc, queryloc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   queryloc       -- Query location address --
                  -- Clause: 7.11.1 --

   notation       -- Query notation --
      NAME        -- Lextype: NOTATION --
      #REQUIRED

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- queryloc -->
```

7.11.2 Name list query

The element form **name list query** (*nmquery*) is a specialized form of query location address for which the query domain must either be the content tree rooted at an SGML element node in an SGML document grove or the value of the entities property of an SGMLDOC node. The attribute **query domain** (*qdomain*) addresses the location source for the query (qdomain is a synonym for locsrc).

```
              <!-- Name List Query Location Address -->
<![ %nameloc; %queryloc; [
<!element
   nmquery        -- Name list query location address --
                  -- Clause: 7.11.2 --
                  -- Locates elements or entities by querying their
                     properties. --
   - O
   (%HyCFC;)*

-- Attributes [locs]: locsrc, nmquery, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nmquery        -- Name list query location address --
                  -- Clause: 7.11.2 --

   HyBase   NAME     #FIXED queryloc
   HyBnames CDATA    #FIXED "locsrc qdomain"

   qdomain        -- Query domain --
      CDATA       -- Reference --
      #IMPLIED    -- Default: implied as described for impsrc --

   notation       -- Query notation --
      NAME        -- Lextype: NOTATION --
      #REQUIRED

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- nmquery -->
```

7.12 Bibliographic location address

The element form **bibliographic location address** (*bibloc*) addresses information objects that systems are not expected to access automatically. A bibliographic location address always returns itself.

NOTE 204 The term “bibliographic” refers to the classic “bibliographic model” of referencing by means of a description; it does not imply that a referenced object must always be a printed document. It could, for example, be a video tape, radio or television broadcast, person, or organization.

NOTE 205 Useful standards may exist for certain forms of query in bibliographic location addresses, even though the located object may not be deliverable by computer. For example, the SMPTE 12M standard for Time and Control Code for Video Tape provides for referencing by reel name, time code number (start-time), and duration. If the object is deliverable by computer, a queryloc can be used with the same form of query.

NOTE 206 An application usually renders a reference to a bibliographic location by presenting the address itself, presumably with the intention that access to the object will be accomplished by the user.

The attribute **bibliographic source** (*bibsrc*) addresses another bibloc element that provides a context for the bibloc.

NOTE 207 For example, a bibloc that is a bibliography entry might have a bibsrc that indicates what floor in a particular library books in its category are on.

Although a bibliographic location address could locate multiple objects, it is not a “multiple location address element” as that term is used in this International Standard.

```
               <!-- Bibliographic Location Address -->
<![ %bibloc; [
<!element
   bibloc         -- Bibliographic location address --
                  -- Clause: 7.12 --
                  -- Bibliographic reference to real object.
                     Represents an address that is processable only as
                     data.  Always returns itself (the bibloc
                     element). --
   - O
   (%HyCFC;)*

-- Attributes [locs]: bibloc --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [locs]: bibloc:bibsrc --
>
<!attlist
   bibloc         -- Bibliographic location address --
                  -- Clause: 7.12 --

   bibsrc         -- Bibliographic source. Another bibloc element that
                     describes the context to which this bibloc is to
                     be applied. --
      CDATA       -- Reference --
                  -- Reftype: bibloc* --
      #IMPLIED
>
]]><!-- bibloc -->
```

8 Hyperlinks module

This clause describes the optional hyperlinks module of HyTime.

8.1 Concepts and definitions

A hyperlink is a typed relationship among two or more objects, each of which fulfills a unique role in the relationship.

NOTE 208 Not all relationships are hyperlinks. Hierarchies (containment), event schedules (coordinate relationships), and simple cross-references (ID/IDREF) are other kinds of relationships that lack some properties of hyperlinks and introduce other properties of their own.

NOTE 209 The term “hyperlink” is used in preference to the unqualified term “link” to avoid confusion with the SGML processing link feature. However, the term “link” can be used with more restrictive qualifying adjectives, as in “hypertext link”, or with no qualifiers when the context is clear.

In interactive applications, hyperlinks typically allow a user to “traverse” the links in arbitrary ways to access information in more than one order. However, HyTime does not require that presentation applications necessarily enable interactive traversal among the anchors of hyperlinks.

NOTE 210 For example, a HyTime application may use hyperlinks merely to control processing of the anchors or may “pre-traverse” hyperlinks by statically presenting one anchor at the point of occurrence of another anchor.

NOTE 211 Although a degree of hyperlinking can be accomplished using the ID and IDREF attribute types, the hyperlinks module offers the following advantages:

a) The hyperlinks module provides a set of architectural forms for the description of links and their traversal rules.

b) The standardization of link traversal and resolution mechanisms makes it possible for a common hyperlinking engine to support multiple applications (although this is not a requirement for HyTime conformance).

c) Not all IDREFS are intended for user interaction or other conditional access; some merely associate information needed for the interpretation or processing of the document (for example, the attribute “subdvl” in *A.5.6 Default value list*). When hyperlink architectural forms are available, they can be used to distinguish true hyperlinks from other uses of IDREF, and to specify traversal rules for interactive use.

NOTE 212 In many cases, it will be possible to make existing SGML documents with cross-references conform to HyTime simply by adding fixed attributes to their attribute definition lists, without modifying the document instance. The refloc facility can be used to adapt hyperlinking element types that do not use ID references to address their anchors.

8.1.1 Link creation

As an abstract object, a hyperlink has three primary properties:

— The “link type”

— The objects addressed as the “anchors” of the hyperlink Anchors can, at designer discretion, be satisfied by lists of multiple objects or by the hyperlink itself.

— The semantic roles the anchors play in the relationship represented by the hyperlink (the “anchor roles”).

NOTE 213 For example, the relationship type “employment” could be defined as having two anchor roles, “employee” and “employer”. An instance of an employment link would address two anchors consisting of single objects: the person who satisfies the employee role and the company (or person) that satisfies the employer role. Relationships from a single person to multiple employers would be represented by allowing the employer anchor to be a list. Relationships from a single employer to multiple employees would be represented by allowing the employee anchor to be a list. Relationships among multiple people and their corresponding employers would be represented by making both the anchors correspondent lists.

A hyperlink is represented by an element whose element type is the link type. The **anchor roles** (*anchrole*) attribute defines the anchor role names. The objects for each anchor role are addressed by attributes whose names are the same as the anchor role names (or mapped to the anchor role names by a HyNames attribute). The element form **hyperlink** (*hylink*) provides the general form for hyperlinks. HyTime also defines specialized forms of hylink that reflect common practice or provide additional linking semantics.

The “link type” of a hyperlink associates a semantic role with the link. Link types are defined by applications, and by architectures to which an application might conform. They are represented by the element type (generic identifier) of each link element. As with all HyTime architectural forms, an application is free to define other attributes for the link type, including attributes that create subcategories of the basic type.

NOTE 214 The decision whether to have multiple element types or a single type with a “subtype” attribute occurs frequently in document type design. Although a “subtype” attribute has the benefit of reducing the number of GIs of which the user must be aware, it adds an equal number of attribute values. Other significant considerations are whether the choice of subtype places constraints on the values of other attributes, or on the content model. If subtypes tend to differ in these respects, the designer should consider creating separate element types, or possibly application-level architectural forms.

The definition of a link type also defines the number of anchors and associates semantic roles with them. Applications can also define semantics for the content of hyperlinks, whether or not they name themselves as an anchor.

NOTE 215 For example, an application can define the content of a particular link type to be an explanation of each instance of the link.

NOTE 216 The designation of anchor roles is not a constraint on the anchor element types. Application designers can enforce such a constraint, if desired, by use of the reftype attribute.

NOTE 217 Although the number of anchors of a hyperlink is constrained, the number of objects that can be linked is not, as any (or all) anchors may be lists (if allowed by the definition of the link type).

NOTE 218 As representations of typed relationships, hyperlinks are not distinguished by the form of location address used to address their anchors. In other words, whether the anchors of a link are addressed by ID, tree location, or query does not effect the relationship established among them. However, the form of address used may present practical constraints on some processors. For example, an Internet-based application may not support the use of queries for initiation anchors of hyperlinks because it is impractical to resolve the query before presenting the anchors. Applications can also change the form of address without affecting the meaning of a hyperlink. For example, documents authored with links that use queries to address the anchors could be prepared for optimized delivery by resolving the queries and replacing them with direct addresses to the objects returned by the query without affecting the hyperlinks themselves.

8.1.2 Link traversal

The addresses of hyperlink anchors will, on request of the application, be resolved by the HyTime engine in order to provide access to the anchors.

NOTE 219 For example, hyperlink elements could represent index entries, for which an application could request access automatically and all at once in order to format the index.

In some applications, particularly interactive ones, hyperlinks are used to represent decision points that affect access to parts of the hyperdocument, represented by the anchors of the hyperlink. Such applications require *selective* resolution of anchor addresses and access to the anchors. Selective access of an anchor of a hyperlink is called “traversal”. An application can specify the traversal that is permitted with respect to each anchor by means of the **link traversal rules** (*linktrav*) attribute.

NOTE 220 In HyTime, traversal is defined by the hyperlinks module. However, the attendant resolution of location addresses that is required in order to locate the selected anchor is an aspect of object identification and is defined in the base and location address modules.

Traversal occurs only among anchors or between adjacent members of list anchors, never to or from the link element itself (unless the link is also an anchor of itself). An anchor that can be accessed from outside the link and from which traversal within the link is possible is called an “initiation anchor”. The classical “bidirectional link” is one that has two anchors, both of which are initiation anchors.

NOTE 221 An application can provide methods of access to link elements and their anchors that are not “traversal” as defined in this International Standard.

When a link is not one of its own anchors or is not an initiation anchor, it is termed independent, as it is contextually independent of its anchors. In particular, the link element may be in a separate document from all of its anchors. From the point of view of their anchors, independent links are “invisible” except by their effect on the anchors themselves. However, HyTime applications may make the properties of hyperlinks, including any content of independent hyperlink elements, available on request.

NOTE 222 In addition, applications may choose to provide facilities for accessing the anchors of links when accessing the link element themselves, even if the links do not name themselves as anchors. In other words, an application may choose to consider that every link is always one of its own anchors for the purposes of enabling movement to the link element itself (but not for the purpose of describing relationships). However, such behavior is not required by HyTime.

When a hyperlink names itself as one of its own anchors and is also an initiation anchor it is said to be “contextual” because the link presumably can occur in some reader-accessible context from which it is expected to be used as an initiation anchor. The contextual link (clink) element form represents the typical use of contextual links to create simple cross references.

When an anchor is a list or correspondent list, it can have an associated list traversal rule that defines how or if traversal is to be be allowed from one member of the list to other members (“list traversal”). The list is treated as a linked list, possibly circular. List traversal can be directional, either left or right, or bidirectional. If the list is considered to be circular, traversal from the first node to the last, the last to the first, or both, is allowed (depending on the directionality of the list).

NOTE 223 For example, if a list anchor allows “adjacent list traversal”, the presentation system could provide “previous” and “next” buttons to reflect the list traversal options, as well as providing a mechanism for traversing from a member of the list to a different anchor of the same hyperlink.

When two or more anchors of a hyperlink are correspondent list anchors, traversal from one such anchor to another is always from a node in one correspondent list to the corresponding node in another such anchor. Correspondent list anchors may also have list traversal rules.

NOTE 224 For example, a relationship such as "defensive match-up" between the members of one sports team and the corresponding players on another sports team could be represented by correspondent anchors, one for each team.

When the scheduling module is supported, it is possible to specify positions and sizes, in both time and space, for the rendition of accessed anchors and previews by using anchloc location addresses to address links or their anchors as the content of scheduled events (see *8.3.2 Hyperlink anchor location address*).

8.1.3 Traversal Rules

The attribute form **hyperlink traversal rules** (*traverse*) provides attributes for controlling traversal among hyperlink anchors and among the members of list anchors.

The attribute **link traversal rules** (*linktrav*) specifies for each anchor whether or not it is an initiation anchor and what traversal is allowed following access of the anchor following traversal to it from another anchor of the same link ("internal traversal"). The basic traversal options are shown in the following list. Some options can be combined.

For the purposes of discussing anchor traversal, the terms "arrival", "departure", and "traversal" have the following specialized meanings:

traversal — The movement from one anchor of a hyperlink to another anchor of the same hyperlink. Traversal can be visualized as moving "through" the link to get from one of its anchors to another. Traversal is always with respect to a single link, regardless of whether or not other links may use the same objects as anchors.

arrival — The arrival at an anchor of a hyperlink by means other than traversal of the link under consideration. For example, in a typical online presentation, scrolling a document to the point where an anchor occurs would constitute arrival at the anchor. Arrival is always "external" to the link. In particular, traversal to an anchor that happens to also be the anchor of another link would constitute arrival at the anchor with respect to the second link.

departure — Moving from an anchor of a link to a place that is not an anchor of the same link. For example, having traversed to an anchor, scrolling somewhere else would constitute departure. Departure is always with respect to the link used to traverse to the anchor. In particular, if two links share the same object as an anchor, following traversal to that object via the first link, traversal of the second link would constitute departure with respect to the first link.

NOTE 225 In other words, given a link of two anchors, anchor A and anchor B, a reader would be saxid to "arrive at" anchor A, "traverse to" anchor B, then "depart from" anchor B.

The basic traversal options are:

— Traversal following (external) arrival ("E"). The anchor is an initiation anchor such that, having arrived at the anchor, traversal is possible to any of the other anchors of the link that do not otherwise prohibit traversal to them. The "E" option does not imply the ability to depart the anchor upon return or traversal to the anchor.

  NOTE 226 External arrival does not constitute having traversed to the anchor.

— Return ("R"). Can only return to the anchor from which traversal to this anchor occurred.

  NOTE 227 For example, when traversed to, a return anchor might be presented in a pop-up window that will not allow any other actions until the pop-up window is dismissed. In addition, the typical "back" function provided by many hypertext browsers is not "return" as defined here, as the invocation of such a facility does not usually relate to the anchor directly. In other words, a strict return must be invoked by directly interacting with the anchor traversed to.

— Internal (“I”). Having traversed to this anchor of the link, can traverse to any other anchor of the same link.

— Departure (“D”). Having traversed to this anchor, can depart the anchor and the hyperlink.

> NOTE 228 For example, when traversed to, the document view might simply be “scrolled” to the anchor, the reader left free to scroll or otherwise navigate to other locations outside the scope of the anchor.

— No further traversal (“N”). Having traversed to this anchor, no further traversal is possible.

> NOTE 229 For example, for hyperlinks that represent decision points in a simulation, “N” anchors could represent decisions that result in termination of the simulation.

— Traversal is prohibited (“P”). Traversal to the anchor is prohibited from any other anchor of the hyperlink.

If an anchor for which departure is *not* allowed is the anchor of another hyperlink or contains anchors of other hyperlinks, traversal of those links is not allowed when the anchor has been traversed to because to do so would constitute departure of the anchor. However, if the anchor is traversed to as an anchor that allows departure or is accessed by means other than hyperlinking, then the departure restriction is not in effect.

NOTE 230 In other words, a hyperlink can affect movement to or from an anchor only with respect to the interpretation of that hyperlink. By the same token, designers and authors can expect their traversal choices to be respected, to the degree that a processing system supports traversal rules at all. (However, the specification of particular traversal rules cannot guarantee access to or restriction of access to hyperlink anchors. In other words, the traversal rule specification represents author or designer intent but cannot guarantee implementation of that intent.)

NOTE 231 Applications may associate data access policies with data objects based on their status as anchors of hyperlinks irrespective of traversal status or activity. However, such policies are beyond the scope of HyTime's traversal rules (although the HyTime activity tracking policy facility may be used to define such policies and their associations with data objects and hyperlinks — see *6.7.3 Activity policy association*).

These basic traversal options can be combined in specific combinations, forming the keywords allowed for the linktrav attribute as follows:

I	traversal after traversal from an anchor of the same link
R	return after traversal
D	departure after traversal
A	any traversal or departure (equivalent to EID)
N	no traversal after traversal from an anchor of the same link
P	traversal to this anchor is prohibited
ID	traversal or departure
RD	Return or departure
EI	traversal after arrival, traversal after traversal from an anchor of the same link
ER	traversal after arrival, return after traversal from an anchor of the same link
ED	traversal after arrival, departure after traversal from an anchor of the same link
EN	traversal after arrival, no traversal after traversal from an anchor of the same link
EP	traversal after arrival, traversal to the anchor prohibited (that is, return prohibited)

ERD traversal after arrival, return, or departure (The ERD option differs from the A option only when there are more than two anchors of the hyperlink).

The value of the linktrav attribute is either one keyword for each anchor or one keyword for all anchors.

NOTE 232 Traversal rules more complex than those expressible by the linktrav keywords can be implemented by normalizing all links to binary links or by defining application-specific traversal specification mechanisms.

The attribute **list traversal** (*listtrav*) applies to any list or correspondent list anchor and defines, for each anchor, the list traversal allowed. Nodes in a list anchor are listed in the order they are addressed. Except for the “N” (no traversal) option, the list is treated as a linked list. When “wrap” is allowed, the list is a circular linked list such that the first node is linked to the last node.

NOTE 233 For the purposes of link traversal, each member of a list anchor is treated as a separately-traversable object. Thus the typical behavior of a browser upon a request to traverse from one anchor of a link to the other anchors would be to present a list showing all of the anchors, including the individual members of any list anchors (perhaps grouped to indicate their membership in a single anchor). For correspondent list traversal, traversal from a member of a correspondent list is always to the corresponding member of a corresponding list anchor.

The possible list traversal options are:

N No list traversal possible. Having traversed to a member of the list, the only possible option is to traverse to another anchor of the hyperlink (which indirectly allows traversal to another member of the same anchor).

NOTE 234 In other words, there is no next or previous behavior, but the other members of the anchor can be traversed to by selecting them from the list presented when link traversal is requested.

L Left traversal. Traversal is from a node in the list to the preceding member of the list.

R Right traversal. Traversal is from a node in the list to the following member of the list.

A Adjacent traversal. Traversal is from a node in the list to either of the adjacent nodes.

LW Left with wrap: Leftward traversal, wrapping to the end of the list when traversing from the first node in the list.

RW Right with wrap: Rightward traversal, wrapping to the beginning of the list when traversing from the last node in the list.

AW Adjacent with wrap: Traversal in either direction, wrapping at either end of the list.

The value of the listtrav attribute is either one for each list anchor or one for all list anchors.

```
                    <!-- Traversal Attributes -->
<![ %traverse; [
<!attlist
-- traverse --    -- Traversal attributes --
                  -- Clause: 8.1.3 --
   (agglink,hylink,ilink)

   linktrav       -- Hyperlink traversal rules --
                  -- Traversal between anchors of hyperlinks:
                       A  any traversal or departure (EID)
                       D  departure after internal arrival
                       E  traversal after external arrival
                       I  traversal after internal arrival
                       N  no traversal after internal arrival
                       P  no internal arrival
                       R  return traversal after internal arrival --
      NAMES       -- Lextype: ("A"|"EI"|"ER"|"ED"|"EN"|"EP"|"ERD"|
                                "I"|"ID"|"D"|"N"|"P"|"R"|"RD")+ --
                  -- Constraint: one per anchor or one for all --
      A

   listtrav       -- List traversal rules --
                  -- Traversal between members of list anchors:
                       A  adjacent (both left and right) traversal
                       L  left traversal
                       N  no traversal
                       R  right traversal
                       W  wrapping traversal --
      NAMES       -- Lextype: ("A"|"AW"|"L"|"LW"|"N"|"R"|"RW")+ --
                  -- Constraint: one per list anchor or one for all
                     list anchors --
      N
>
]]><!-- traverse -->
```

8.2 Hyperlink architectural forms

HyTime defines the following element forms that an application can use to represent arbitrary link types:

— Hyperlink (hylink) is the most flexible. It can have any number of anchors and may be completely independent of any of its anchors.

— Contextual link (clink) represents simple, generic cross references.

— Aggregate link (agglink) represents aggregation relationships.

— Variable link (varlink) represents hyperlinks in which the anchor roles may vary among instances of the same link type.

— Independent link (ilink) provides an alternative syntax for hyperlinks in which the anchor addresses are specified with a single IDREFS attribute, rather than with separate attributes for each anchor role.

8.2.1 Hyperlink

The element form **hyperlink** (*hylink*) is a hyperlink whose location in a document need have no relation to the location of its anchors. The significance of the location (if any) is determined by the application.

NOTE 235 A hylink can be used in situations where it is not possible to modify the anchors (perhaps because there is no write access). If there are two anchors and one can be modified, it may be more convenient to use a contextual link.

NOTE 236 Link types that could be independent links include external thesauri and indexes, indicators of various kinds of alternative versions, and graphical user interface objects, such as menus and radio button sets.

NOTE 237 One useful application of hyperlinks that could be modelled on the hylink form is known as a “property link”. It allows a value of a specified property type to be associated with a location, as when a collaborative author or reviewer is annotating a document. There are two anchors: the “subject”, which identifies the location with which the property is being associated, and the “property value”, which identifies the value of the property. The traversal rules must at least permit traversal to begin from the subject and return from the property value, although a designer could choose to make the link bidirectional.

The attribute **anchor roles** (*anchrole*) identifies the number of anchors by specifying a role name for each. The same role name cannot be used for more than one anchor (enabling addressing of anchors by anchor role name).

The number of anchors can exceed two only if the “manyanch” option is supported and is specified with no value or a value greater than “2”. The anchor roles must be constant in a client document. Each anchor role name can be followed by either of the following keywords:

#LIST The anchor is a “list anchor”: the anchor may be a node list of one or more nodes.

#CORLIST The anchor is a “correspondent list anchor”: the anchor may be a node list of one or more nodes. In addition, for each member of a correspondent list anchor, there must be a corresponding member in all the other correspondent list anchors of the same hyperlink. If there is only one #CORLIST anchor, it is treated as a normal #LIST anchor.

Anchors that are not list anchors are “singleton anchors” and may only resolve to a single node.

For each anchor role (except self anchors), there must be an attribute whose name is the name of an anchor role and that is a referential attribute. Each such attribute addresses, directly or indirectly, the members of the anchor with that role. These attributes are referred to generically as the “anchor addressing attributes” of the link.

NOTE 238 The declared value prescription for the attribute must either be IDREF, IDREFS, ENTITY, ENTITIES or the form of location address used by the attribute must be specified through the refloc facility (see *7.8 Reference location address*).

It is an RHE to address multiple objects for a singleton anchor.

NOTE 239 The names of attributes used to address the anchors of a hyperlink can be mapped to names other than the anchor role names using the HyNames architectural renaming attribute. For example, the following HyNames specification maps the anchor role name “employees” to the attribute name “empIDs”:

```
HyNames NAME #FIXED "employees empIDs"
```

This remapping will always be recognized by a HyTime processor but will be ignored by a generic architecture processor when the attributes are not also declared in an architectural meta-DTD (as would be the case if an element is derived directly from the hylink form, rather than from an architectural form derived from the hylink form).

NOTE 240 An application designer can constrain the element types of the anchors by means of a reftype attribute.

The attribute **anchor existence constraints** (*anchcstr*) defines for each anchor whether or not the anchor can be omitted or whether or not the link element itself is to be taken as the anchor if no anchor-addressing attribute is specified for the anchor. The attribute value is a list of keywords, one for each anchor or one for all. The anchor constraints must be constant for a given link type in a client document.

The existence constraints for an anchor apply when an anchor-addressing attribute is not specified or the attribute value is the empty string (“”). Attributes that have a default value other than #IMPLIED, #CONREF, or an empty string are still considered to be specified if they do not occur on the element's start-tag.

NOTE 241 There is a distinction between not specifying an anchor-addressing attribute and addressing an empty node list. An anchor for which an empty node list is addressed is validated according to the rules specified by the *emptyanc* attribute.

The possible anchor existence constraint keywords are:

REQUIRED Required anchor: The anchor is required and cannot be the link itself. The anchor-addressing attributes for required anchors normally have a literal default value or a default value prescription of #REQUIRED or #FIXED. By default all anchors are required.

SELF Self anchor: If no anchor-addressing attribute is specified for the anchor, the hyperlink itself is the anchor. The anchor-addressing attributes for self anchors normally have a default value prescription of #IMPLIED or are not declared at all (in which case the hyperlink is always the anchor).

OMIT Omissible anchor: If no anchor-addressing attribute is specified for the anchor, the anchor is taken to not exist for the link: the role is omitted from the instance of the relationship. The anchor-addressing attributes for omissible anchors normally have a default value prescription of #IMPLIED.

COND Conditional link anchor: If no anchor-addressing attribute is specified for the anchor, the anchor is taken to be omitted (as for OMIT). In addition, if all the anchors of a link are conditional and no anchor-addressing attributes are specified or only one anchor remains, then the element is treated as though its HyTime architectural form were HyBrid: it is not considered to be a link but other architectural processing is performed. The anchor-addressing attributes for conditional link anchors normally have a default value prescription of #IMPLIED.

The attribute **is empty anchor an error?** (*emptyanc*) specifies, for each anchor, whether it is an error for the anchor to be an empty list (“ERROR”) or not (“NOTERROR”). The value is either one keyword for each anchor or one for all anchors. The empty anchor rule only applies to anchors for which anchor-addressing attributes have been specified, either on an element start-tag or because the attributes have default values other than #IMPLIED or #CONREF.

NOTE 242 In other words, empytanc does not apply to omitted OMIT or COND anchors.

All links must have at least two anchors. It is an RHE for a link to have zero or one anchors (unless the COND value of the anchcstr attribute is specified such that the lack of sufficient anchors causes the element to not be a hyperlink at all).

NOTE 243 For a binary link, for example, it would be an error to have one self anchor and one omissible anchor and then omit the anchor, as the link would only have one anchor (the self anchor). However, if the omissible anchor were instead a conditional link anchor, it would not be an error to omit the anchor because when omitted, the element would not be taken to be a link.

```
                   <!-- Hyperlink Relationship -->
<![ %hylink; [
<!element
   hylink         -- Hyperlink relationship --
                  -- Clause: 8.2.1 --
   - O
   (%HyCFC;)*

-- Attributes [links]: hylink --
-- OptionalAttributes [links]: traverse --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   hylink         -- Hyperlink relationship --
                  -- Clause: 8.2.1 --

   anchrole       -- Anchor roles  --
      CDATA       -- Lextype: (NAME,("#LIST"|"#CORLIST")?)+ --
                  -- Constraint: all #CORLIST anchors of a link must
                     have the same number of members. --
      #REQUIRED   -- Constant --

   anchcstr       -- Anchor existence constraints --
      NAMES       -- Lextype: ("REQUIRED"|"SELF"|"OMIT"|"COND")+ --
                  -- Constraint: one per anchor or one for all --
      REQUIRED    -- Constant --
   emptyanc       -- Is empty anchor an error? --
      NAMES       -- Lextype: ("ERROR"|"NOTERROR")+ --
                  -- Constraint: one per anchor or one for all --
      ERROR

-- Anchor addressing attributes: one per non-self anchor role, with
   same name as anchor role (unless remapped by the architectural
   attribute renamer attribute). --
>
]]><!-- hylink -->
```

8.2.2 Contextual link

The element form **contextual link** (*clink*) represents the most common form of generic cross reference. The clink form is derived from the hylink form.

NOTE 244 Any hyperlink may, potentially, be “contextual” by using itself as one of its anchors and making that anchor a traversal initiating anchor.

The link type of a contextual link expresses the type of reference. There are two anchor roles: the “reference mark” (named “REFMARK”), which is the link element, and the “reference subject” (named “REFSUB”), which is identified by the attribute **reference subject** (*refsub*). The reference subject can be a list.

NOTE 245 The contextual link is an optimization of the configuration of hylink that occurs most frequently in conventional documents; it can model such common reference elements as footnote references and figure references. If more complex configurations are wanted (such as more anchors), the hylink element form can be used.

The reference mark can be empty if it is simply a point in the document, rather than an object.

NOTE 246 Contextual link elements can use the common opacity attribute to indicate that they are “transparent” in the context in which they occur. See *A.5.2 Common attributes of elements*.

```
                       <!-- Contextual Link -->
<![ %clink; [
<!element
   clink          -- Contextual link --
                  -- Clause: 8.2.2 --
   - O
   (%HyCFC;)*

-- Attributes [links]: clink --
-- OptionalAttributes [links]: clinktra --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   clink          -- Contextual link --
                  -- Clause: 8.2.2 --

   HyBase   NAME     #FIXED hylink
   HyBnames CDATA    #FIXED "refsub linkend"
   anchrole CDATA    #FIXED "refmark refsub #LIST"
   anchcstr NAMES    "self required"

   linkend        -- Reference subject link end --
```

```
      CDATA       -- Reference --
      #IMPLIED    -- Default: omitted or conditional --
                  -- Constraint: if left unspecified, must specify
                     COND in anchcstr --
>
]]><!-- clink -->

                  <!-- Contextual Link Traversal -->
<![ %clink; %traverse; [
<!attlist
-- clinktra --    -- Contextual link traversal --
                  -- Clause: 8.2.2 --
   (clink)

   linktrav CDATA    "A ID"
>
]]><!-- clink, traverse -->
```

8.2.3 Aggregation link

The element form **aggregation link** (*agglink*) represents the grouping of a list of nodes into an “aggregate”. The aggregate represents the list of members as a whole, providing the semantic that the list of members can be treated as a single object.

NOTE 247 In particular, aggregate links can be members of list anchors, enabling the representation of “nested” lists or cascading menus.

An agglink has two anchor roles, the “aggregate” (agg) and “members of the aggregate” (members).

NOTE 248 An aggregation link is not a location address. Therefore a reference to an agglink element is always a reference to the element itself, not the members of the members anchor.

```
                      <!-- Aggregation Link -->
<![ %agglink; [
<!element
   agglink        -- Aggregation link --
                  -- Clause 8.2.3 --
   - O
   (%HyCFC;)*

-- Attributes [links]: agglink
-- OptionalAttributes [links]: traverse --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   agglink        -- Aggregation link --
                  -- Clause 8.2.3 --

   HyBase   NAME     #FIXED hylink
   anchrole CDATA    #FIXED "agg members #LIST"
   anchcstr NAMES    "self required"
   loctype  CDATA    "members IDLOC"
   members         -- Members of aggregate --
      CDATA        -- Reference --
                  -- Note: Because members is by default given the
                     location address type IDLOC, its value is
                     interpreted by default as if it were IDREFS,
                     though the interpretation can be changed by
                     specifying a different value for the loctype
                     attribute. --
      #IMPLIED     -- Default: omitted or conditional --
                  -- Constraint: if left unspecified, must specify
                     COND in anchcstr --
>
]]><!-- agglink -->
```

8.2.4 Variable link

The element form **variable link** (*varlink*) represents hyperlinks in which the anchor roles need not be constant for a given link type.

Rather than individual anchor-addressing attributes, the varlink form uses **anchor specification** (*anchspec*) subelements to define each anchor role and address the members of that anchor. The content of the anchspec element contains the address of the anchor. The attribute **anchor role** (*anchrole*) specifies the anchor role name. If no anchrole attribute is specified, the element type of the anchspec element is used as the anchor role. If multiple anchspec elements in a single varlink exhibit the same anchor role, the nodes they address are combined to form a single list of members for that anchor, listed in the order the anchspec elements occur within the varlist. The number of anchors can exceed two only if the “manyanch” option is supported and is specified with no value or a value greater than “2”.

The attribute **may anchor have multiple members?** (*multmem*) indicates whether or not the anchor allows multiple members (list), multiple members that correspond to members in other anchors (corlist), or only a single member (single). If multiple anchspec elements in a single varlink have the same anchor role, they must specify the same value for multmem, either “list” or “corlist”.

The other attributes are as defined for the hylink element form.

The varlink element may be a self anchor, indicated by specifying a value for the anchrole attribute of the varlink element form.

For the purposes of associating anchor roles with link types, the link type associated with a given varlink element type is the union of all anchor role names used for that element type in a single document (e.g., as though all the anchor roles had been explicitly declared and given an anchor existence constraint of “OMIT”).

```
            <!-- Variable Link -->
<![ %varlink; [
<!element
   varlink         -- Variable link --
                   -- Clause: 8.2.4 --
   - O
   (anchspec)+

-- Attributes [links]: ancspcat, varlink, vartrav --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
```

```
>
<!element
   anchspec       -- Anchor specification --
                  -- Clause: 8.2.4 --
   - O
   (%HyCFC;)*     -- Reference --
                  -- Note: Use of refloc facility required --

-- Attributes [links]: anchspec, ancspcat, vartrav --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   anchspec       -- Anchor specification --
                  -- Clause: 8.2.4 --

   multmem        -- May anchor have multiple members? --
      (single|list|corlist)
      single

   emptyanc       -- Is empty anchor an error? --
      (error|noterror)
      error
>

<!attlist
-- ancspcat --    -- Anchor specification attributes --
                  -- Clause: 8.2.4 --
   (anchspec,varlink)

   anchrole       -- Anchor role --
      NAME
      #IMPLIED    -- Default: for anchspec, anchor role is GI of element --
                  -- Default: for varlink, varlink is not self anchor --
>
<![ %traverse; [
<!attlist
-- vartrav --     -- Varlink traversal rules --
                  -- Constraint: ignored on varlink element if it is
                     not a self anchor --
   (anchspec,varlink)

   linktrav       -- Hyperlink traversal rules --
                  -- Traversal between anchors of hyperlinks:
                       A  any traversal or departure (EID)
                       D  departure after internal arrival
                       E  traversal after external arrival
                       I  traversal after internal arrival
                       N  no traversal after internal arrival
                       P  no internal arrival
                       R  return traversal after internal arrival --
      (A|EI|ER|ED|EN|EP|ERD|I|ID|D|N|P|R|RD)
      A
   listtrav       -- List traversal rules --
                  -- Traversal between members of list anchors:
                       A  adjacent (both left and right) traversal
                       L  left traversal
                       N  no traversal
                       R  right traversal
                       W  wrapping traversal --
      (A|AW|L|LW|N|R|RW)
                  -- Constraint: ignored if anchor is not a list --
      N
>
]]><!-- traverse -->
]]><!-- varlink -->
```

8.2.5 Independent link

The element form **independent link** is a hyperlink that uses a single IDREFS attribute to address its anchors. The ilink form is semantically identical to hylink but provides an alternative syntax for addressing its anchors.

The attribute **anchor roles** (*anchrole*) identifies the number of anchors by specifying a role name for each. The same role name cannot be used for more than one anchor. The anchor roles must be constant for a given link type in a client document.

The keyword “#AGG” is a synonym for “#LIST”.

NOTE 249 The “#AGG” keyword is provided for existing ilink elements.

The number of anchors can exceed two only if the “manyanch” option is supported and is specified with no value or a value greater than “2”.

Each ID specified by the linkends attribute is associated with the corresponding anchor role name specified by the anchrole attribute, in the order specified in the source document.

NOTE 250 An application designer can constrain the element types of the anchors by means of a reftype attribute.

The ilink element itself can be designated as an anchor by specifying the element's ID as a link end or by specifying exactly one fewer IDs than there are anchor roles, in which case the ilink element is taken to be its own first anchor (equivalent to using an anchor existence constraint of SELF for the first anchor).

```
                      <!-- Independent Link -->
<![ %ilink; [
<!element
   ilink          -- Independent link --
                  -- Clause: 8.2.5 --
   - O
   (%HyCFC;)*

-- Attributes [links]: ilink --
-- OptionalAttributes [links]: traverse --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   ilink          -- Independent link --
```

```
                  -- Clause: 8.2.5 --

   anchrole       -- Anchor roles --
      CDATA       -- Lextype: (NAME,("#AGG"|"#LIST"|"#CORLIST")?)+ --
                  -- Constraint: one per anchor --
      #REQUIRED   -- Constant --

   linkends       -- Link ends --
      IDREFS      -- Reference --
                  -- Constraint: One ID per anchor role, except
                     that the first ID may be omitted if the first
                     anchor is the link element itself. --
      #REQUIRED
>
]]><!-- ilink -->
```

8.3 Hyperlink-related location addresses

The following location address forms provide queries for addressing hyperlinks and their anchors. Both forms are derived from the fundamental queryloc element form (see *7.11.1 Query location address*).

8.3.1 Hyperlink location address

The element form **hyperlink location address** (*linkloc*) addresses hyperlinks by link type. The content of the linkloc is a list of zero or more GIs. The location source of the linkloc is the document or documents from which links are to be selected. If no GIs are specified, the linkloc addresses all hyperlinks in the location source documents.

NOTE 251 In effect, linkloc selects from HyTime semantic groves derived from the location source documents all nodes whose class is “hyperlink” and whose “typename” property value is equal to one of the link type names specified (see annex B for details).

```
                 <!-- Hyperlink location address -->
<![ %linkloc; [
<!notation
   HyLnkLoc       -- HyTime hyperlink queryloc notation --
                  -- Clause: 8.3.1 --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" location source is
                     document in which the primary tree root or
                     referrer occur. --
                  -- Lextype: (csname|literal)+ --
                  -- Constraint: csnames and literals must be link
                     types (GIs) in location source. --
                  -- Constraint: if no link types specified, addresses
                     all links in location source. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Hyperlink Queryloc Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!element
   linkloc        -- Hyperlink location address --
```

```
                  -- Clause: 8.3.1 --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" location source is
                     document in which the primary tree root or
                     referrer occur. --
   - O
   (#PCDATA)      -- Lextype: (csname|literal)+ --
                  -- Constraint: csnames and literals must be link
                     types (GIs) in location source. --
                  -- Constraint: if no link types specified, addresses
                     all links in location source. --

-- Attributes [locs]: locsrc, impsrc --
-- Attributes [links]: linkloc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   linkloc        -- Hyperlink location address --
                  -- Clause: 8.3.1 --

   HyBase   NAME     #FIXED queryloc
   notation NOTATION (HyLnkLoc) #FIXED HyLnkLoc

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- linkloc -->
```

8.3.2 Hyperlink anchor location address

The element form **hyperlink anchor location address** (*anchloc*) addresses the anchors of hyperlinks by anchor role name. The content of the anchloc is a list of zero or more anchor role names. If no anchor role names are specified, anchloc addresses all anchors of all hyperlinks in the location source.

The location source of anchloc shall be one or more documents or link elements. For documents, all links in the document are addressed.

NOTE 252 The hyperlink location address form can be used to address all links of a particular type.

NOTE 253 In effect, for each link addressed as the location source, anchloc selects the nodes in the “object” or “members” properties of the anchor or list anchor subnodes whose “rolename” property value matches the anchor role name (see annex B for details).

```
              <!-- Hyperlink Anchor Location Address -->
<![ %anchloc; [
<!notation
   HyAncLoc       -- HyTime hyperlink anchor queryloc notation --
                  -- Clause: 8.3.2 --
                  -- Constraint: location source objects must be
                  hyperlinks or documents --
               -- Constraint: if implied location source is
                  "ptreert" or "referrer" (and referrer is not a
                  hyperlink) location source is document in which
                  the primary tree root or referrer occur. --
               -- Lextype: (csname|literal)* --
               -- Constraint: csnames and literals are anchor
                  role names of links in location source. --
               -- Constraint: if no anchor roles specified,
                  addresses all anchors of links in location
                  source. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Hyperlink Anchor Queryloc Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!element
   anchloc        -- Hyperlink anchor location address --
                  -- Clause: 8.3.2 --
                  -- Constraint: location source objects must be
                     hyperlinks or documents --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" (and referrer is not a
                     hyperlink) location source is document in which
                     the primary tree root or referrer occur. --
   - O
   (#PCDATA)      -- Lextype: (csname|literal)* --
                  -- Constraint: csnames and literals are anchor
                     role names of links in location source. --
                  -- Constraint: if no anchor roles specified,
                     addresses all anchors of links in location
                     source. --

-- Attributes [locs]: locsrc, impsrc --
-- Attributes [links]: anchloc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   anchloc        -- Hyperlink anchor location address --
                  -- Clause: 8.3.2 --

   HyBase   NAME     #FIXED queryloc
   notation NOTATION (HyAncLoc) #FIXED HyAncLoc

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
```

```
<!entity % impsrc "INCLUDE">
]]><!-- anchloc -->
```

9 Scheduling module

This clause describes the optional scheduling module of HyTime. It requires use of the coordinate addressing facilities described in *6.8 Coordinate Specifications*.

9.1 Scheduling concepts and definitions

There can be no concept of “alignment” or “synchronization” (which is alignment in time) of objects without the more basic concept of “scheduling” them; that is, giving them a position and size. Scheduling, in turn, requires a set of coordinate axes and a system for measuring on them, collectively called a “coordinate space”. In HyTime, a coordinate space has a finite extent along each axis and is known as a “finite coordinate space” (FCS).

NOTE 254 An FCS is also a “finite coordinate space” in the mathematical sense that it has a finite number of axes or dimensions. In mathematical terminology it is also a discrete coordinate space because each axis has a finite number of points.

An object has no inherent extent or position in a coordinate space. It is given them by incorporating the object in an “event”, which is positioned in a list of events, called an “event schedule”. A coordinate space can have many event schedules, each of which is aligned with the coordinate axes.

The positioning of an event in a schedule is one of the values specified by the event's “scheduled extent”. A scheduled extent consists of dimension specifications, one for each of the axes of the coordinate space, which collectively define both the position and the literal extent of the event.

Within an event schedule, logical groups of events can be combined into an “event group” in order to associate common properties with them.

In HyTime, each axis of a coordinate space is assigned to a measurement domain that is based on a standard measurement unit (SMU), such as the Systeme International second, or the Systeme International meter. An application can also define virtual measurement units that have no physical basis (that is, are not “real”). These are figures of merit that can represent relative extents. During rendition (see *10 Rendition module*) of the document, virtual measures can be projected onto any real measurement domain that is appropriate, or onto another virtual domain.

NOTE 255 For example, at rendition time a virtual unit could be assigned a fixed value, an algorithmically generated variable value, or, in the time domain, be related to a click track recorded by a performer.

NOTE 256 If convenient, a virtual unit could be thought of as corresponding to a unit of real measurement. Such a convention would not preclude the creation of other renditions made according to other methods.

NOTE 257 Axes need not be temporal or spatial; the quanta can denominate any measurable thing, for example money, temperature, and mass, making the FCS concept broadly applicable to the visualization of information. Some applications may even use an axis to impose an arbitrary order on some set of topics, one quantum per topic, as seen in certain kinds of tabular information. In the latter case, depending on the topics themselves and the significance of the order thus imposed, it may or may not be meaningful to say that (topic one) + 2 = (topic three).

9.2 Measurement units

At the root of any standardized measurement system there must be a base that is not open to conflicting interpretations. In HyTime, that base is called a “standard measurement unit” (SMU). In the introduction to coordinate addressing (*6.8 Coordinate Specifications*), the SMU was a “generic quantum” — a virtual measurement suitable only for expressing relative sizes and positions. However, when the scheduling module is used, there are also real and other virtual SMUs. The real SMUs are based on Systeme International (SI) units, such as the SI second and the SI meter.

In order to guarantee consistency in the meaning of SMUs, they are declared with formal public identifiers in notation declarations. SMUs can be declared by applications, but HyTime provides an initial set (see *9.2.3.1 Common Standard Measurement Units*).

9.2.1 Measurement domain definition

An SMU is unlikely to be the correct granularity for all hypermedia applications, or even for all extent specifications in a single application. It is therefore possible to create a "measurement domain definition" that establishes a set of base granules, all defined ultimately in terms of the SMU of that domain. For further flexibility, "HyTime Measurement Units (HMUs)" can be used by defining them as a ratio of a granule.

NOTE 258 For example, HMUs with such values as 24 and 30 per second can be used for synchronization with motion pictures and video.

The element form **measurement domain definition** (*measure*) establishes a set of base granules, each defined as a ratio of a "reference granule", and all defined ultimately in terms of the SMU of the domain.

NOTE 259 For example, a measurement domain based on the SI second could include granules for "hour" and "microsecond".

The attribute **domain SMU** (*SMU*) specifies the standard measurement unit for the domain. It must have been declared as the notation name of a data content notation with a formal public identifier that is a measurement unit. If the domain is virtual, the text identifier component of the public identifier must be "Virtual Measurement Unit".

Multiple measurement domain definitions with the same domain SMU are treated as defining a single measurement domain.

NOTE 260 This rule makes it easy to supplement a standard measurement domain definition with granules needed for a specific document.

Instances of the element form **granule definition** (*granule*) define granules in terms of the domain SMU. For each instance, the attribute **granule name** (*gn*) names the granule, and the content of the element is its definition. The definition is expressed in the notation specified by the attribute **granule definition notation** (*gdnot*).

Granule names are case-sensitive.

NOTE 261 For example, "mm" and "Mm" are two distinct granule names in the SI meter domain.

A granule name cannot be the same as another granule name in the same measurement domain.

A granule name cannot be the same as the SMU name of its measurement domain. As granule names are case-sensitive and SMU names are not, case is ignored when applying this rule.

NOTE 262 For example, the granule name "sisecond" is not permitted in the measurement domain "SIsecond" because the granule name and SMU name are the same when case is ignored.

```
                <!-- Measurement Domain Definition -->
<![ %measure; [
<![ %HyGrand; [
   <!entity % dgdnot "HyGrand">
]]>
<!entity %
   dgdnot         -- Default granule definition notation --
                  -- Clause: 9.2.1 --

   "#REQUIRED"
>
```

```
<!element
   measure        -- Measurement domain definition --
                  -- Clause: 9.2.1 --
                  -- Defines a set of granules in terms of a standard
                     measurement unit (SMU).  Multiple measurement
                     domain definitions for the same SMU comprise one
                     domain. --
   - O
   (granule+)

-- Attributes [sched]: measure --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   measure        -- Measurement domain definition --
                  -- Clause: 9.2.1 --

   smu            -- Standard measurement unit --
                  -- The standard measurement unit for the domain --
      NAME        -- Lextype: SMU --
      #REQUIRED
>
<!element
   granule        -- Granule definition --
                  -- Clause: 9.2.1 --
                  -- The content of a granule definition element
                     defines the mathematical relationship between the
                     granule being defined and the measurement domain's
                     SMU. --
                  -- Note: The relationship may be defined in terms of
                     another granule of the same measurement domain. --
   - O
   (%HyCFC;)*

-- Attributes [sched]: granule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   granule        -- Granule definition --
                  -- Clause: 9.2.1 --

   gn             -- Granule name --
                  -- The granule name being defined. Granule names are
                     not normalized as SGML names (that is, they are
                     case-sensitive). --
      CDATA       -- Lextype: granule --
                  -- Constraint: unique within measurement domain --
                  -- Constraint: cannot be same as SMU name (with case
                     ignored) --
      #REQUIRED
   gdnot          -- Granule definition notation --
                  -- Notation used to specify the relationship between
                     the granule being defined and the measurement
                     domain's SMU. --
      NAME        -- Lextype: NOTATION --
      %dgdnot;
>
]]><!-- measure -->
```

9.2.2 HyTime granule definition notation

By default, the **HyTime granule definition notation** (*HyGrand*) is used to define granules.

A HyGrand expression defines a granule (that is, a quantum type) as a ratio of it to a “reference” granule. The ratio is specified as a pair of numbers, the first being the numerator and the second being the denominator. The reference granule can be any granule that was defined without reference (direct or indirect) to the granule being defined.

```
             <!-- HyTime Granule Definition Notation -->
<![ %HyGrand; [
<!notation
   HyGrand        -- HyTime Granule Definition Notation --
                  -- Clause: 9.2.2 --
                  -- The mathematical relationship between a granule
                     and its measurement domain's SMU is specified as
                     a ratio of the granule being defined to either
                     the SMU or another granule of the same domain. --
                  -- Lextype: (ratio,(granule|SMU)) --
                  -- Constraint: A granule may not be defined in terms of
                     itself, either directly or indirectly. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Granule Definition Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!entity % measure "INCLUDE">
]]><!-- HyGrand -->
```

9.2.3 Useful measurement domains

HyTime includes measurement domain definitions for virtual time (virTime), virtual space (virSpace), real time (SIsecond), and real space (SImeter), in addition to an explicit domain definition for the generic quantum (gQuantum) (see *9.2.3.2 Measurement domain definitions*).

NOTE 263 Differences in granule names can be meaningful to an application, even when they are equivalent to HyTime (for example, “thumb” and “inch” in the SImeter domain).

NOTE 264 The wide range of granules allows convenient choices for diverse applications.

NOTE 265 Real measurement in a computer is not a continuum. There is a discrete number of computer clock ticks per second and a discrete number of points of resolution on a display or printer. This fact simplifies measurement calculations for HyTime.

9.2.3.1 Common Standard Measurement Units

The following notation declarations provide Standard Measurement Units required for most hypermedia applications.

```
                      <!-- Measurement Units -->
<![ %sched; [
<!notation
   gQuantum       -- Generic quantum --
                  -- Reference unit of generic quanta --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Virtual Measurement Unit//EN"
>
<!notation
   SIsecond       -- Systeme International second --
                  -- Reference unit of real time --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International second//EN"
>
<!notation
   SImeter        -- Systeme International meter --
                  -- Reference unit of real length --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International meter//EN"
>
<!notation
   virTime        -- Virtual Time --
                  -- Reference unit of virtual time --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Virtual Measurement Unit//EN"
>
<!notation
   virSpace       -- Virtual Space --
                  -- Reference unit of virtual space --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Virtual Measurement Unit//EN"
>
]]><!-- sched -->
```

9.2.3.2 Measurement domain definitions

HyTime defines five measurement domain definitions for the useful SMU notation declarations, that can be referenced as public text using the following SGML formal public identifier:

```
"ISO/IEC 10744:1997//TEXT
 Useful Measurement Domain Definitions//EN"
```

The element and attribute list declarations for “measure” and “granule” are not included here. They are identical to the definitions of the architectural forms, except that the content of “granule” is declared to be #PCDATA.

NOTE 266 As the set of granules is unbounded, these measurement domain definitions are necessarily incomplete. Therefore, the absence of a granule should not be construed as a rejection. Granules can be added easily using the facilities described in *9.2.1 Measurement domain definition*. For example, granules could be added to represent clock ticks for various computer systems.

9.2.3.2.1 Generic quantum

```
<measure smu=gQuantum>
  <granule gn=quantum>               1      1 gQuantum
  <granule gn=bit-combination>       1      1 quantum
  <granule gn=token>                 1      1 quantum
  <granule gn=node>                  1      1 quantum
  <granule gn=cell>                  1      1 quantum
</measure>
```

9.2.3.2.2 Virtual time

```
<measure smu=virTime>
  <granule gn=vtu>                   1      1 virTime
  <granule gn=tact>                  1      1 vtu
  <granule gn=metronom>              1      1 vtu
</measure>
```

9.2.3.2.3 Virtual space

```
<measure smu=virSpace>
  <granule gn=vsu>                   1      1 virSpace
</measure>
```

9.2.3.2.4 Systeme International second

```
 <measure smu=SIsecond>
  <granule gn=ysec>                  1   1000 zsec
  <granule gn=zsec>                  1   1000 asec
  <granule gn=asec>                  1   1000 fsec
  <granule gn=fsec>                  1   1000 psec
  <granule gn=psec>                  1   1000 nsec
  <granule gn=nsec>                  1   1000 usec
  <granule gn=usec>                  1   1000 msec
  <granule gn=msec>                  1     10 csec
  <granule gn=csec>                  1     10 dsec
  <granule gn=dsec>                  1     10 second
  <granule gn=SMPTE-50>              1     50 second
  <granule gn=V1250>                 1     50 second
  <granule gn=NTSC>                  1     30 second
  <granule gn=SMPTE-240M>            1     30 second
  <granule gn=SMPTE-30>              1     30 second
  <granule gn=SMPTE-30-drop>       100   2997 second
  <granule gn=SMPTE-25>              1     25 second
  <granule gn=PAL>                   1     25 second
  <granule gn=SECAM>                 1     25 second
  <granule gn=European>              1     25 second
  <granule gn=SMPTE-24>              1     24 second
  <granule gn=motion-picture>        1     24 second
  <granule gn=SMPTE-24-drop>       100   2396 second
  <granule gn=PC-tick>              10    182 second
  <granule gn=second>                1      1 SIsecond
  <granule gn=dasec>                10      1 second
  <granule gn=hsec>                 10      1 dasec
  <granule gn=ksec>                 10      1 hsec
  <granule gn=Msec>               1000      1 ksec
  <granule gn=Gsec>               1000      1 Msec
  <granule gn=Tsec>               1000      1 Gsec
  <granule gn=Psec>               1000      1 Tsec
  <granule gn=Esec>               1000      1 Psec
  <granule gn=Zsec>               1000      1 Esec
  <granule gn=Ysec>               1000      1 Zsec
  <granule gn=minute>               60      1 second
  <granule gn=quarter-hour>         15      1 minute
  <granule gn=half-hour>            30      1 minute
  <granule gn=hour>                 60      1 minute
  <granule gn=day>                  24      1 hour
  <granule gn=week>                  7      1 day
  <granule gn=fortnight>             2      1 week
 <!-- Year and the granules based on it might not be accurate enough
      for some applications. -->
  <granule gn=year>            3652422  10000 day
  <granule gn=decade>               10      1 year
  <granule gn=century>              10      1 decade
  <granule gn=millennium>         1000      1 year
 </measure>
```

9.2.3.2.5 Systeme International meter

```
<measure smu=SImeter>
  <granule gn=ym>                    1   1000 zm
  <granule gn=zm>                    1   1000 am
  <granule gn=am>                    1   1000 fm
  <granule gn=fm>                    1   1000 pm
  <granule gn=pm>                    1   1000 nm
  <granule gn=angstrom>              1     10 nm
  <granule gn=nm>                    1   1000 um
  <granule gn=um>                    1   1000 mm
  <granule gn=mm>                    1     10 cm
  <granule gn=cm>                    1     10 dm
  <granule gn=dm>                    1     10 meter
  <granule gn=meter>                 1      1 SImeter
  <granule gn=dam>                  10      1 meter
  <granule gn=hm>                   10      1 dam
  <granule gn=km>                   10      1 hm
  <granule gn=Mm>                 1000      1 km
  <granule gn=nautical-mile>      1852      1 meter
  <granule gn=Gm>                 1000      1 Mm
  <granule gn=AU>                    1 206265 parsec
  <granule gn=Tm>                 1000      1 Gm
  <granule gn=lightyear>           100    326 parsec
  <granule gn=Pm>                 1000      1 Tm
  <granule gn=parsec>             3086      1 Tm
  <granule gn=Em>                 1000      1 Pm
  <granule gn=Zm>                 1000      1 Em
  <granule gn=Ym>                 1000      1 Zm
  <granule gn=microinch>             1   1000 milliinch
  <granule gn=milliinch>             1   1000 inch
<!-- Point and pica are the rounded versions commonly used in computer
     systems and might not be accurate enough for some applications. -->
  <granule gn=point>                 1     12 pica
  <granule gn=pica>                  1      6 inch
  <granule gn=barleycorn>            1      3 inch
  <granule gn=nail>                  1     16 yard
  <granule gn=inch>                254    100 cm
  <granule gn=thumb>                 1      1 inch
  <granule gn=hand>                  4      1 inch
  <granule gn=Roman-foot>         1164    100 inch
  <granule gn=foot>                 12      1 inch
  <granule gn=Greek-foot>         1244    100 inch
  <granule gn=Northern-foot>        11     10 foot
  <granule gn=cubit>                18      1 inch
  <granule gn=Sumerian-cubit>      495     10 cm
  <granule gn=royal-cubit>        2062    100 inch
  <granule gn=yard>                  3      1 foot
  <granule gn=ell>                   5      4 yard
  <granule gn=fathom>                6      1 foot
  <granule gn=perch>                11      2 yard
  <granule gn=pole>                  1      1 perch
  <granule gn=rod>                   1      1 perch
  <granule gn=chain>                66      1 feet
  <granule gn=furlong>             220      1 yard
  <granule gn=quarter-mile>        440      1 yard
  <granule gn=mile>               1760      1 yard
  <granule gn=league>                3      1 mile
</measure>
```

9.2.3.3 Other standard measurement units

The following notation declarations can be used for virtual reality applications or other situations that require units of measure other than temporal, spatial, and generic quanta. They comprise the remaining Systeme International base units and supplementary units.

```
<![ %sched; [
<!notation
   SIkg           -- Systeme International kilogram --
                  -- Reference unit of mass --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International kilogram//EN"
>
<!notation
   SIkelvin       -- Systeme International kelvin --
                  -- Reference unit of thermodynamic temperature --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
```

```
           Systeme International kelvin//EN"
>
<!notation
   SIcd           -- Systeme International candela --
                  -- Reference unit of luminous intensity --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International candela//EN"
>
<!notation
   SIampere       -- Systeme International ampere --
                  -- Reference unit of electric current --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International ampere//EN"
>
<!notation
   SImole         -- Systeme International mole --
                  -- Reference unit of amount of substance --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International mole//EN"
>
<!notation
   SIradian       -- Systeme International radian --
                  -- Reference unit of plane angle --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International radian//EN"
>
<!notation
   SIsr           -- Systeme International steradian --
                  -- Reference unit of solid angle --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International steradian//EN"
>
]]><!-- sched -->
```

Declarations for other SI derived units should be constructed similarly: the notation name should consist of the letters “SI” followed by the special name of the derived unit if it is 6 characters or less, otherwise by the symbol for the derived unit. (The special name and the symbol can be found in ISO 31-0:1991.) For example:

```
<![ %sched; [
<!notation
   SIC            -- Systeme International degree Celsius --
                  -- Reference unit of celsius temperature --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International degree Celsius//EN"
>
]]><!-- sched -->
```

Declarations of primary units of national currency should be constructed as follows: the notation name should consist of the nation's three character country code as specified in ISO 3166, followed by the first five characters of the name of the currency. For example:

```
<![ %sched; [
<!notation
   CHNYUAN        -- Yuan currency unit of China --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Yuan currency unit of China//EN"
>
<!notation
   GBRPOUND       -- Pound currency unit of United Kingdom --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Pound currency unit of United Kingdom//EN"
>
]]><!-- sched -->
```

9.3 Finite coordinate space

Each **finite coordinate space** (*fcs*) element corresponds to a distinct logical FCS. Each fcs element defines a set of coordinate axes, each of which is associated with a single measurement domain. An FCS with more than one axis requires support of the “manyaxes” option with no value specified, or with a value greater than 1.

NOTE 267 Throughout this international standard, the lowercase name “fcs” means “finite coordinate space element”, and the uppercase “FCS” means the logical (that is, semantic) finite coordinate space represented as nodes in a grove after all HyTime processing has been completed.

The **FCS axis names** (*axes*) attribute defines the axes of the FCS by defining a name for each axis and associating with it an SMU that identifies the measurement domain of the axis. The value of the attribute is a list of axis name/ notation name pairs, one for each axis.

An fcs element governs associated elements that use the “sched” attribute form. Event schedules, wands (which require the object modification facility), and batons (which require the event projection facility) become associated with an fcs element by being contained in it and/or by referring to it with their fcs attributes.

Axis names must be unique within the axes attribute's value. Client fcs elements must declare for each axis named an “axis dimension specification” attribute whose name is the same as an axis named by the axes attribute (or is mapped to an axis name by the HyNames architectural renaming attribute).

HyTime engines schedule extents in terms of “measurement domain units” (MDUs). The MDU is the finest internal granularity on the basis of which all calculations will be made.

NOTE 268 The MDU is normally the common denominator of all HMUs used to schedule extents on a given axis. MDU granularity should be fine enough that roundoff errors will not be significant (that is, under normal circumstances, perceivable).

Each axis dimension specification attribute has the lexical type:

```
(NUMBER, granule?, ("#MDU", ratio, granule?)?)
```

The number specifies the dimension of the axis (ultimately, the total number of MDU quanta along the axis), expressed in terms of the following granule, if specified; otherwise, the units of the dimension used are the SMU specified for the axis. If a granule is specified, it is both the default base granule and default HMU for the axis; otherwise the default base granule and HMU is the axis SMU.

NOTE 269 The “default base granule” for an axis is the granule used to express dimensions (in HMUs) along it in, for example, events scheduled in evscheds in which an axis attribute does not specify a granule to which the HMU is proportional. The default HMU for an axis is the HMU used to express dimensions in, for example, events scheduled in evscheds in which an axis attribute for the axis does not exist, or does not specify a ratio of HMUs to default granules.

If #MDU and a ratio is specified, it defines the MDU to SMU ratio for the axis, specified in terms of either the following granule or the default HMU. All granule names must be defined in terms of the axis SMU.

The number of MDUs on any axis must not exceed the number derived from the value of the HyTime architecture use declaration hyqcnt attribute for the document in which the fcs element occurs.

NOTE 270 The aspects of a finite coordinate space defined by HyTime in the fcs architectural form do not fully define a space. However, application designers can use them in conjunction with application semantics to do so.

```
                   <!-- Finite Coordinate Space -->
<![ %sched; [
<!element
   fcs            -- Finite coordinate space --
                  -- Clause: 9.3 --
   - O
   (baton|evsched|wand)*

-- Attributes [sched]: fcs --
-- OptionalAttributes [sched]: calibrat --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, pdimref:prjtarg,
   rfcsloc:prjsrc, sched:fcs --
>
<!attlist
   fcs            -- Finite coordinate space --
                  -- Clause: 9.3 --

   axes           -- Axis names and measurement domains --
                  -- The name and measurement domain (SMU) of each
                     axis. --
      CDATA       -- Lextype: (AXISNM,SMU)+ --
                  -- Constraint: Each axis name must be unique
                     within the list. --
      #REQUIRED   -- Constant --

-- Each axis name specified in the value of the axes attribute is
   taken to be the name of an attribute of the client element that
   specifies the dimension and granularity of the axis.  Each such
   attribute has the lexical type (NUMBER, granule?, ("#MDU", ratio,
   granule?)?), where the number is the dimension of the axis,
   expressed in terms of the following granule, if specified;
   otherwise the dimension is in terms of the axis SMU.  If a granule
   is specified, it is both the default base granule and default HMU
   for the axis; otherwise the default base granule and HMU is the
   axis SMU.  If #MDU and a ratio is specified, it defines the MDU to
   SMU ratio for the axis, specified in terms of either the following
   granule or the default HMU.  All granule names must be defined in
   terms of the axis SMU. --
>
]]><!-- sched -->
```

9.3.1 Axis calibration

One or more axes of a finite coordinate space may be calibrated with respect to an external context. The external context need not be finite in dimension or resolution; its meaning is unknown to a HyTime engine. The external context should be appropriate to the measurement domains of their corresponding axes.

NOTE 271 For example, it would make sense to peg a point on an axis measured real time to a particular historical (or future) date, or a point on an axis measured in degrees Celsius to the boiling point of water at one bar of atmospheric pressure. It would not be appropriate to calibrate an axis measured in Australian dollars in terms of a geographic meridian.

Because the external context with respect to which an axis is calibrated is of potentially infinite resolution, the calibration must take place at a point along the axis rather than at a quantum. A calibration point is defined by specifying a quantum number and whether the point occurs at the beginning point of the quantum (“STARTS”), or at end point of the quantum (“ENDS”).

The attribute **axis calibration** (*calibrat*) specifies one or more axis name, attribute, notation tuples, one tuple for each axis to be calibrated The attribute name is the name of an attribute of the client element whose value determines the position of the axis with respect to an external context. The notation names the coordinate specification notation used by the attribute named in the tuple (e.g., the HyTime Calendar Specification Notation, HyCalSpc).

The values of client attributes named in the calibrat attribute have an unnormalized lexical type of (s*, granule?, s+, snzi, s+, (“STARTS”|“ENDS”) #ORDER SGMLCASE, s, char*). The first part of the attribute value (up to the last s separator) specifies the calibration point for the axis. The optional granule (“granule?”) is the granule in which the quantum number is expressed. If a granule name is not specified, the default granule for the axis is assumed. The quantum number on the axis, a signed non-zero integer (snzi), is specified as if it were the first marker in a marker pair used to specify a dimension; the second number of the pair is understood always to be 1 (only one quantum is ever specified). Either “starts” or “ends” must be specified. (The case of these tokens is not significant, as indicated by the “#ORDER SGMLCASE” HyLex specification that follows it.) “Starts” means that the calibration point is at the beginning of the specified quantum; “ends” means that it is at the end of it.

The rest of the value (that which matches char*) is a position within an external context, and is specified in the notation specified following the attribute's name in the axis calibration attribute.

NOTE 272 Real time axes can be calibrated using the HyCalSpc notation defined by this international standard (see *9.9.1 HyTime calendar specification notation*). For example, “1972-01-01 CE” is data in HyCalSpc notation meaning “January 1 of the year 1972 of the common era”. The axis calibration value “year -1 ends 1972-01-01 CE” means that the last year on the axis would conclude just at the point in real time when 1972 began. Any notation could be used, of course, and the point in real time could be expressed with whatever level of precision is supported by the notation.

```
                      <!-- Axis calibration -->
<![ %calibrat; [
<!attlist
-- calibrat --    -- Axis calibration --
                  -- Clause: 9.3.1 --
   (fcs)

   calibrat       -- Axis calibration --
                  -- The calibration of an axis is defined by
                     specifying the name of the axis followed by the
                     name of an attribute of the client element whose
                     value determines the position of the axis with
                     respect to an external context. --
      CDATA       -- Lextype: (AXISNM,ATTNAME,NOTATION)+ --
                  -- Constraint: Each axis name may appear only
                     once. --
                  -- Constraint: Calibration points should be
```

```
                     expressed in notations which make sense, given
                     the measurement domains of their corresponding
                     axes. --
      #IMPLIED    -- Default: no axis calibration --

-- Attributes named by the axis calibration attributes have an
   unnormalized lexical type of:

   (s*,granule?,s+,snzi,s+,("STARTS"|"ENDS") #ORDER SGMLCASE,s,char*)

   The first part of the attribute value (up to the last s separator)
   specifies the calibration point for the axis.  The rest of the
   value (that which matches char*) is a position within an external
   context, and is specified in the notation given following the
   attribute's name in the axis calibration attribute. --
>
<!entity % sched "INCLUDE">
]]><!-- calibrat -->
```

9.4 Scheduling and extents

A dimension specification represents a coordinate location on a single axis (see *6.8 Coordinate Specifications*). When the scheduling module of HyTime is used, finite coordinate spaces with multiple axes can be defined.

An “extent specification” represents a coordinate location or “bounding box” on all of the axes of a finite coordinate space and consists of one dimension specification per axis.

The set of first quanta of each dimension specification of an extent defines the position of the bounding box. The set of quantum counts for all the dimension specifications defines its size.

NOTE 273 The bounding box need not be rectangular, as HyTime makes no assumptions as to the relationship, if any, between the coordinate axes. However, as HyTime's principal field of application deals with coordinate spaces that can be rendered perceivable by humans, use of terminology that implies orthogonal axes seems justified.

HyTime offers an application a construct for collecting and aligning extents called a “schedule”. A schedule allows the application to define the order in which dimensions will be specified (that is, the order of the axes).

9.4.1 Schedules

A schedule is a list of elements, each having a dimension on all axes of the coordinate space in which the schedule occurs or to which it refers. HyTime defines three kinds of schedules: event schedules, wands, and batons; these elements use the **schedule** (*sched*) attribute form.

The attribute **governing fcs element** (*fcs*) refers to the fcs element that defines the logical FCS within which the scheduled elements occur. If the fcs attribute is not specified, the schedule must be a subelement of an fcs element and is a schedule of the logical FCS corresponding to that fcs element.

The attribute **axis order** (*axisord*) specifies the order of axes for extent specification within the schedule if they are different from the order the axes are specified by the axes attribute of the governing fcs element. If at least one axis name is specified, omitted axis names mean all elements in the schedule fully occupy the omitted axes. For each axis name specified, an attribute of the same name may be used to specify the HMU for the axis. The lexical type of such attributes is (ratio, granule?), where the ratio is the number of HMUs per base granule, and the granule, if specified, is the base granule. If a granule is not specified, the default base granule defined by the governing fcs element is used. A specified granule must be defined in terms of the axis SMU. If a value is not supplied for an axis HMU specification attribute, the HMU for that axis is the default HMU defined by the governing fcs element.

The attribute **sorted** (*sorted*) specifies whether the order of the SGML representation of the elements in the schedule is sorted. If so, elements with multiple scheduled extents must have their extents sorted, and the elements are sorted by their first extents. The sort order is by position on the axes, in the order specified by the axis order attribute, with the first specified axis being the primary sort key, the second axis the secondary key, and so on.

The attribute **extent coverage** (*coverage*) specifies whether or not gaps may occur between the scheduled extents of elements in the schedule.

The attribute **extent overlap** (*overlap*) specifies whether or not scheduled extents of elements in the schedule may overlap. If the schedule is a wand, the value of the overlap attribute is ignored and treated as if it were “noverlap”.

NOTE 274 The extents of the modscopes on a given wand are never permitted to overlap.

```
                          <!-- Schedule -->
<![ %sched; [
<!attlist
-- sched --        -- Schedule --
                   -- Clause: 9.4.1 --
   (baton,evsched,wand)

   fcs             -- Governing finite coordinate space element --
      CDATA        -- Reference --
                   -- Reftype: fcs --
      #IMPLIED     -- Default: containing fcs element --
                   -- Constraint: required if schedule is not contained
                      by fcs element. --

   axisord         -- Axis order --
                   -- The order of axes for extent specification within
                      schedule.  For each axis name specified, an
                      attribute of the same name may be used to specify
                      the HMU for the axis.  The lexical type of such
                      attributes is (ratio, granule?), where the ratio
                      is the number of HMUs per base granule, and the
                      granule, if specified, is the base granule.  If a
                      granule is not specified, the default base
                      granule defined by the governing fcs element is
                      used.  A specified granule must be defined in
                      terms of the axis SMU.  If a value is not
                      supplied for an axis HMU specification attribute,
                      the HMU for that axis is the default HMU defined
                      by the governing fcs element. --
      CDATA        -- Lextype: (AXISNM)+ --
                   -- Constraint: axis names must be defined by
                      governing fcs element --
      #IMPLIED     -- Constant --
                   -- Default: Axis order is unchanged from that
                      specified by governing fcs element, and each axis
                      may have an HMU specification attribute. --

   sorted          -- Extent ordering --
                   -- Ordering of elements with respect to the ordering
                      of their scheduled extents. --
      (sorted|unsorted)
      unsorted
   coverage       -- Extent coverage --
                  -- Whether or not gaps may occur between the
                     scheduled extents of elements in the schedule. --
      (solid|sparse)
      sparse

   overlap        -- Extent overlap --
                  -- Whether or not scheduled extents of elements in
                     the schedule may overlap. --
                  -- Constraint: If the schedule is a wand, the value
                     is ignored and is always treated as noverlap. --
      (overlap|noverlap)
      #IMPLIED    -- Default: If the schedule is not a wand, the
                     default value is overlap. --
>
]]><!-- sched -->
```

9.4.2 Extent specification

The attribute form *exspec* consists of the single attribute **nominal extent specification** (*exspec*), which associates a scheduled extent (or extents) with an element. Multiple scheduled extents can be specified if it is desired to represent the cloning or repetition of the element in more than one position and/or in different sizes.

NOTE 275 The semantic of having multiple extents for a single event is not defined. This means that an application can be defined in such a way that the object scheduled by an event is somehow divided among the extents, or that the object is duplicated, with one whole copy for each extent, or some combination of the two. There are many other possibilities, as well, with many possible refinements.

```
                    <!-- Extent Specification -->
<![ %sched; [
<!attlist
-- exspec --      -- Nominal extent specification --
                  -- Clause: 9.4.2 --
   (event,modscope,proscope)

   exspec         -- Extent specification --
      CDATA       -- Reference --
                  -- Reftype: (extent|extlist)+ --
      #REQUIRED
>
]]><!-- sched -->
```

9.4.3 Group extent specification

Events, modscopes, and proscopes can appear within the syntactic container elements evgrp, modgrp, and progrp, respectively. These container elements can appear within evscheds, wands, and batons, respectively, along with events, modscopes, and proscopes that are not contained in evgrps, modgrps, and progrps. Moreover, evgrps, modgrps, and progrps are recursive. The semantic significance of containment within groups is primarily defined by the application. However, the grpdex, repscope, and axis-named attributes of the **group derived extent specification** (*grpdex*) and **group repetition specification** (*grprepet*) attribute forms, which are used only in the evgrp, modgrp, and progrp element forms, carry HyTime-defined semantics which can be used to make the effective extents of the elements contained by them proportional to, and/or translations of, their nominal extents (grpdex), to specify an area in which the group is repeated (repscope), and to specify other criteria governing repetition and the calculation of the “first” component of a dimension of the syntactically succeeding event, modscope, and proscope, if any, that is expressed using the implicit dimref feature.

The attribute **group derived extent specification** (*grpdex*), if specified, specifies one or more derived extents of the group. A subelement with more than one nominal scheduled extent will be given a derived extent for each scheduled extent.

Multiple derived extents can be used to represent repetitions with varying proportions and translations: the group's subelements are positioned in the FCS in each of the derived extents. The application defines the semantics applied when events have multiple distinct extents, e.g., whether the object of the event is duplicated among all the extents, or whether it is somehow distributed among them. However, each distinct extent of a modscope or proscope is treated as if it were a distinct modscope or proscope, with the entire modifier or projector brought to bear by each one.

If a value is specified for the grpdex attribute, for each axis, a matrix calculation is performed for each nominal dimension of each nominal extent against each dimension of each extent specified by the grpdex attribute, in which a set of derived effective dimensions is calculated for each nominal dimension of each extent of each contained element by calculating a derived first quantum and quantum count according to the following formulas:

```
DerivedFirstQuantum = GrpdexFirstQuantum +
                      (((NominalFirstQuantum - GroupScopeFirstQuantum)
                        * GrpdexQcnt) / GroupScopeQcnt)
DerivedQcnt = (NominalQcnt * GrpdexQcnt) / GroupScopeQcnt
```

where DerivedFirstQuantum and DerivedQcnt are the “first” and “qcnt” components, respectively, of each derived dimension; NominalFirstQuantum and NominalQcnt are the “first” and “qcnt” components, respectively, of each nominal dimension of each extent of each contained element; GroupScopeFirstQuantum and GroupScopeQcnt are the “first” and “qcnt” components, respectively, of the dimension of the “group scope” (the sum of the extents of the subelements of the group, less any overlap and including any gaps, or, in other words, on each axis, the dimension bounded by the first quantum of the child element with the lowest first quantum, at one end, and the last quantum of the child element with the highest last quantum, on the other end); and GrpdexFirstQuantum and GrpdexQcnt are the “first” and “qcnt” components, respectively, of each dimension of each extent specified by the grpdex attribute. (The asterisks and slashes are multiplication and division operators, respectively.) The resulting dimensions are rounded to the nearest whole MDU.

If no value is specified for the grpdex attribute, the nominal extents of the contained elements are the effective scheduled extents.

NOTE 276 If a value is specified for the grpdex attribute, and it is desired to retain the nominal scheduled extent of each member in addition to the derived extents, an extent equal to the “group scope” extent must be specified as one of the grpdex extents. (The “group scope” extent is the sum of the extents of the subelements of the group, less any overlap and including any gaps, or, in other words, on each axis, the dimension bounded by the first quantum of the child element with the lowest first quantum, at one end, and the last quantum of the child element with the highest last quantum, on the other end.)

If a value is specified for the **repetition scope** (*repscope*) attribute, it is the area within which the contained events, modscopes or proscopes are repeated, subject to additional criteria in the value of each axis-named attribute, if any. For each axis, repetition starts at the first quantum of the repetition scope and continues until the repetition scope is exhausted on each axis.

NOTE 277 Repscope can be used to translate the dimensions of the contained elements.

NOTE 278 The effect of repetition is like repeating patterns on wallpaper or the squares of a chessboard. The repetition of patterns along each axis is not necessarily a single row of repetitions; it may be a row of columns of repetitions.

If no value is specified for the repscope attribute, the first or only occurrence of the group is wherever the effective extent of the group places it, and the number of repetitions is limited only by the repeat count limit parameters of the axis-named attributes.

The grprepet attribute form also has optional attributes, one for each axis and having the same name as the axis for which it specifies repetition parameters. Each such attribute is CDATA #IMPLIED, and conforms to the lexical model:

```
(uint, (("PART"|"NOPART"), unzi)?, unzi?))
```

where the unsigned integer (uint) is the “repeat count limit” parameter, the specification of “part” or “nopart” (“PART”|“NOPART”) is the “allow partial dimensions” parameter, the following unsigned nonzero integer (unzi) is the “start point within pattern” parameter, and the final unsigned nonzero integer (unzi) is the “offset to next occurrence” parameter.

If the value of the **repeat count limit** parameter is 0, there is no repeat count limit. The group is repeated on this axis from the beginning to the end of the dimension specified by the repscope attribute. If no value is specified for the repscope attribute, the 0 value is treated as if it were 1. If the value of the “repeat count limit” parameter is greater than 0, the number of repetitions is limited to the number specified. If no value is specified for the repscope attribute, the number of repetitions is limited only by the number of repetitions specified and the remaining length of the axis. If no value is specified for an axis-named attribute, the “repeat count limit” parameter is treated as if it were 1.

If the value of the **allow partial dimensions** parameter is PART, partial repetitions at the beginning and/or end of the series on this axis will be used to completely fill the repetition scope. If the value is NOPART, partial repetitions do not occur. As a result, there may be unoccupied space at the beginning and/or end of the series on this axis within the repetition scope. If no value is specified for this parameter, the group is treated as if the value were “NOPART” for this axis.

The value of the **start point within pattern** parameter is the number of the HMU within the effective group dimension (where the “first” component of the group's dimension on this axis is considered to be 1) at which the first occurrence in the series begins. If the value is not 1, the remaining partial portion of the group will constitute the first repetition in each series on this axis. This first partial repetition will occur as skipped space if the “allow partial dimensions” parameter's value is “NOPART”.

The value of the **offset to next occurrence** parameter is the number of HMUs that will be treated as the effective qcnt of the group only for purposes of determining the location of the next contiguous repetition, or, in the case of the last or only occurrence of the group, the next dimension with an implicit dimref used as its “first” component. The default value is the effective qcnt of the group.

```
             <!-- Group Derived Extent Specification -->
<![ %grpdex; [
<!attlist
-- grpdex --      -- Group derived extent specification --
                  -- Clause: 9.4.3 --
   (evgrp,modgrp,progrp)

   grpdex         -- Group derived extent specification --
                  -- Use for resizing, rearrangement, repetition.
                     Group members are given the derived extents.  If
                     nominal extent is also wanted, it must be
                     specified as one of the derived extents. --
      CDATA       -- Reference --
                  -- Reftype: (extent|extlist)* --
      #IMPLIED    -- Default: grpdex is same as overall "group scope",
                     that is, the nominal extents are effective --
>
<!entity % sched "INCLUDE">
]]><!-- grpdex -->
```

```
               <!-- Group Repetition Specification -->
<![ %grprepet; [
<!attlist
-- grprepet --    -- Group repetition specification --
                  -- Clause: 9.4.3 --
   (evgrp,modgrp,progrp)

   repscope       -- Repetition scope --
                  -- Scope within which the contained events,
                     modscopes or proscopes are repeated, subject to
                     additional criteria in the value of each
                     axis-named attribute, if any.  For each axis,
                     repetition starts at the first quantum of the
                     repetition scope and continues until the
                     repetition scope is exhausted on each axis. --
       CDATA      -- Reference --
                  -- Reftype: extent --
       #IMPLIED   -- Default: The first or only occurrence of the
                     group is wherever the effective extent of the
                     group places it, and the number of repetitions is
                     limited only by the repeat count limit parameters
                     of the axis-named attributes. --

-- Additional attributes may be defined for client elements, one for
   each axis of the FCS.  The names of the attributes are the axis
   names, and the values of the attributes are specifications
   regarding repetitions of the group and the implicit dimref of the
   succeeding extent along an axis.  Each of the attributes has the
   lexical model (uint,(("NOPART"|"PART"),unzi)?,unzi?), which is
   interpreted as follows:

      "repeat count limit" parameter (uint):
         If the value is 0, there is no repeat count limit.  The group
         is repeated on this axis from the beginning to the end of the
         dimension specified by the repscope attribute.  If no value
         is specified for the repscope attribute, the 0 value is
         treated as if it were 1.  If the value of the "repeat count
         limit" parameter is greater than 0, the number of repetitions
         is limited to the number specified.  If no value is specified
         for the repscope attribute, the number of repetitions is
         limited only by the number of repetitions specified and the
         remaining length of the axis.  If no value is specified for
         an axis-named attribute, the "repeat count limit" parameter
         is treated as if it were 1.

      "allow partial dimensions" parameter ("PART"|"NOPART"):
         If the value is "PART", partial repetitions at the beginning
         and/or end of the series on this axis will be used to
         completely fill the repetition scope.  If the value is
         "NOPART", partial repetitions do not occur.  (There may be
         unoccupied space at the beginning and/or end of the series on
         this axis.)  The default value is "NOPART".

      "start point within pattern" parameter (penultimate unzi):
         The value is the number of the HMU (where the "first"
         component of the group's dimension on this axis is 1) at
         which the first occurrence in the series begins.  If the
         value is not 1, the remaining partial portion of the group
         will constitute the first repetition in each series.  This
         first partial repetition will occur as skipped space if the
         "allow partial dimensions" parameter's value is "NOPART".
         The default value is 1.

      "offset to next occurrence" parameter (final unzi):
         The value is the number of HMUs that will be treated as the
         qcnt of the group for purposes of determining the location of
         the next contiguous repetition (or the next dimension with an
         implicit dimref used as its "first" component).  The default
         value is the effective qcnt of the group. --
>
<!entity % sched "INCLUDE">
]]><!-- grprepet -->
```

9.4.4 Scheduled extent

The element form **scheduled extent** (*extent*) defines the position and size of elements in schedules. By default, the HyTime extent specification notation is used.

NOTE 279 HyTime treats an extent as a resource, which means that it can occur freely in a document and can be referenced by an ID in the contexts in which it is needed. An application could therefore define “named extent” element types that represent, for example, historical periods, and build a resource pool of “period” elements. An attribute of the period elements would be the name of the period, which an application could use for display or other purposes. Events could be positioned with respect to the periods by means of dimension references (see *9.8 Dimension referencing*). Such a pool of period elements would comprise what measurement theory calls an “ordinal scale”. Named extents can also be used to construct ordinal scales in domains other than time; for example, shades of color on an axis that represents a portion of the visible spectrum.

NOTE 280 Designers can also require that extent specifications occur in the content of events by using the refloc facility to address the extent specification of an event by a tree address relative to the event element itself. For example, if the client document type sets the content model of event-form elements to “(extent, (%event-content;)*)”, the tree position “1 1” specified for the exspec attribute of the event element addresses the extent element that is its first subelement:

```
<!ELEMENT event
   - - (extent,(%event-content;)*)
>
<!ATTLIST event
   HyTime   NAME     #FIXED event
   loctype  CDATA    #FIXED "exspec treeloc"

   -- Tree position of extent subelement --
   exspec   NUMBERS  #FIXED "1 1"
>
```

```
            <!-- HyTime Extent Specification Notation -->
<![ %HyExSpec; [
<!notation
   HyExSpec       -- HyTime Extent Specification Notation --
                  -- Clause: 9.4.4 --
                  -- A list of dimensions (in the HyTime Dimension
                     List Notation, after resolution of marker
                     functions) that together specify a single extent
                     on one or more axes. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Dimension Specification Notation//EN"
-- Attributes [sched]: HyExSpec --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyExSpec       -- HyTime Extent Specification Notation --
                  -- Clause: 9.4.4 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyDimLst
>
<!entity % HyDimLst "INCLUDE" >
<!entity % sched "INCLUDE">
]]><!-- HyExSpec -->

                           <!-- Extent -->
<![ %sched; [
<![ %HyExSpec; [
   <!entity % dexspec "HyExSpec">
]]>
<!entity %
   dexspec        -- Default extent specification notation --
                  -- Clause: 9.4.5 --
   "#IMPLIED"
>
<!element
   extent         -- Extent --
                  -- Clause: 9.4.4 --
                  -- A position and quantum count along one or more
                     axes. --
   O O
   (%HyCFC;|%dimlist;)*
                  -- Constraint: if no extent specification notation
                     specified, then content is restricted to
                     (dimspec)* --

-- Attributes [sched]: extent --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, exspec:exspec, grpdex:grpdex,
   grprepet:repscope --
>
<!attlist
   extent         -- Extent --
                  -- Clause: 9.4.4 --

   notation       -- Extent specification notation --
      NAME        -- Lextype: NOTATION --
      %dexspec;   -- Default: content is restricted to (dimspec)* --
>
]]><!-- sched -->
```

9.4.5 Scheduled extent list

The element form **scheduled extent list** (*extlist*) specifies a list of extents within a finite coordinate space.

```
                         <!-- Extent List -->
<![ %sched; [
<![ %HyExtLst; [
   <!entity % dextlist "HyExtLst">
]]>
<!entity %
   dextlist       -- Default extent list notation --
                  -- Clause: 9.4.5 --

   "#IMPLIED"
>
<!element
   extlist        -- Extent List --
                  -- Clause: 9.4.5 --
                  -- A list of extents on one or more axes. --
   O O
   (%HyCFC;|%dimlist;|extent|extlist)*
                  -- Constraint: if no extent list notation
                     specified, then content is restricted to
                     (extent|extlist)* --

-- Attributes [sched]: extlist
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, exspec:exspec, grpdex:grpdex --
>
<!attlist
   extlist        -- Extent List --
                  -- Clause: 9.4.5 --

   notation       -- Extent list notation --
      NAME        -- Lextype: NOTATION --
      %dextlist;  -- Default: content is restricted to
                     (extent|extlist)* --
>
]]><!-- sched -->
```

9.4.6 HyTime extent list notation

The HyTime extent list notation is an extended form of axis marker list in which dimspecs are derived from the axis markers and extents are then derived from the dimspecs. The list can also contain nested extlists and extents, to any level of nesting, but these will merge to create the effect of a single-level list.

The extent list notation processor must be aware of the number of axes included in the governing axis order of the schedules to which an extent specification applies.

Nested extents and extent lists may be included only at extent specification boundaries within the axis marker list.

NOTE 281 In other words, an extent specification cannot appear within what is effectively another extent specification. A nested element can only appear before or after the dimension specifications for a complete cycle of axes.

```
                 <!-- HyTime Extent List Notation -->
<![ %HyExtLst; [
<!notation
   HyExtLst       -- HyTime Extent List Notation --
                  -- A list of extents each of which may be specified
                     as an extent, part of another extent list, or or
                     as all or part of a list of axis markers
                     explicitly entered or returned by a marker
                     function. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Extent List Notation//EN"

-- Attributes [sched]: HyExtLst --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyExtLst       -- HyTime Extent List Notation --
                  -- Clause: 9.4.6 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyDimLst
>
<!entity % HyDimLst "INCLUDE" >
<!entity % sched "INCLUDE">
]]><!-- HyExtLst -->
```

9.5 Event schedule

The element form **event schedule** (*evsched*) is a list of events, each having a dimension on all axes of the coordinate space.

Syntactically contiguous events can occur in “event groups” so that common properties can be associated with them.

There can be more than one event schedule in a coordinate space, and multiple events can potentially exist in the same position, whether in the same or different schedules.

NOTE 282 An application must determine whether multiple events can validly exist in the same position, and the significance of their doing so.

HyTime implies no inherent relationship among event schedules, other than the fact that event schedules that are contained by or refer to the same fcs element exist within the same FCS. Also, when the projection facility is used, batons and baton rules can relate schedules for projection, and wands and wand rules can relate schedules for object modification.

NOTE 283 Other relationships can be defined by applications through the use of element types, attributes, hierarchy, and hyperlinks.

NOTE 284 An application can define several types of event schedule. For example, it could allow different kinds of multimedia event to be intermixed in a generalized “multimedia schedule”, or it could define specialized schedules that are limited to particular event types. An example of a specialized type could be a “storyboard” schedule whose elements are pictures with a specified duration in virtual time, separated by “dissolve” elements of various types, also with durations.

```
                       <!-- Event Schedule -->
<![ %sched; [
<!element
   evsched        -- Event schedule --
                  -- Clause: 9.5 --
   - O
   (event|evgrp)*

-- Attributes [sched]: sched --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, dimref:schdspec,
   pdimref:prjtarg, pulsemap:pulsemap, rfcsloc:prjsrc --
-- Referrers [rend]: batrule:scheds, batrule:targschd, modrule:events,
   prorule:sources, prorule:targschd, wandrule:evscheds --
>
]]><!-- sched -->
```

9.5.1 Scheduled Event

An object has no inherent position in a coordinate space. It is given one by placing it in an “event” (like placing a picture in a frame), which has a dimension on each axis. The set of dimensions is the “scheduled extent” of the event, and it can be said that the event schedules its object by framing it within its scheduled extent.

The element form **scheduled event** (*event*) represents the occurrence of an object within an event schedule. Its extent specification specifies the nominal position and size of that occurrence.

NOTE 285 Due to the nature of some objects, and because of the effect of style sheets and/or the rendering conventions inherent in applications, the extents of the events that schedule objects are not necessarily the same as the extents of the objects as actually rendered. For example, the actual extent occupied by an object may be smaller along some axes, and larger on others, and it may be accurately describable only in terms of many extents which altogether comprise exactly those quanta, along each axis, in which the rendered object appears. This is why it is said that the scheduled extent is “nominal”.

The attribute **event objects** (*object*) is used to designate the object(s) to be scheduled by the event. If left unspecified, the event is its own object.

The attribute **object alignment rule** (*align*) specifies how the object is to be aligned with respect to the extent.

The alignment of an object with an event can be defined on each axis by specifying the name of the axis followed by one of the keywords #LEFT, #RIGHT, or #CENTER. If the keyword is #LEFT, the lowest (leftmost) edge of the object is aligned with the point at the beginning of the lowest quantum of the event. If the keyword is #RIGHT, the highest (rightmost) edge of the object is aligned with the point at the end of the highest quantum of the event. If the keyword is #CENTER, the center point of the object is aligned with the center point of the event.

Alternatively, objects can be aligned with events in much the same way that axes are calibrated with respect to external contexts (see *9.3.1 Axis calibration*), using an object-specific notation. The name of any axis may be followed by the name of an attribute of the client element whose value determines the position of the object with respect to the event. Attributes so named have an unnormalized lexical type of (s*, granule?, s+, snzi, s+, (“STARTS”|“ENDS”) #ORDER SGMLCASE, s, char*). The first part of the attribute value (up to the last s separator) specifies the calibration point within the event. The optional granule (granule?) is the granule in which the quantum number is expressed. If a granule name is not specified, the granule used to express the dimension(s) of the event on the axis is assumed. The quantum number on the axis, a signed non-zero integer (snzi), where 1 is the first quantum of the extent, is specified as if it were the first marker in a marker pair used to specify a dimension; the second number of the pair is understood always to be 1 (only one quantum is ever specified). Either “starts” or “ends” must be specified. (The case of these tokens is not significant, as indicated by the “#ORDER SGMLCASE” HyLex specification that follows it.) “Starts” means that the alignment point is at the beginning of the specified quantum; “ends” means that it is at the end of it. The rest of the value (that which matches char*) is a position within the object and is specified in the notation given following the attribute's name in the object alignment attribute.

HyTime offers several techniques for reusing an event element without repeating its complete specification:

— If there is a semantic implication to the reuse, the extent specification can position the event in several places in an FCS (see *9.4 Scheduling and extents*). Repetitions can also be scheduled using an event group (see *9.4.3 Group extent specification*). In addition, event projection can be used to cause semantic copies of an event to appear in the same and/or other FCSs.

— If there is no semantic implication to the reuse (that is, the application will see the repetitions as independent specifications), an SGML entity reference can be used.

— If the repetitions are the only instances of a unique event type, and the extent is the same for each repetition, it is possible to define the event type as an element type with fixed attribute values. (Example: the company logo.)

— If the repetitions are not the only instances of an event type, it is possible to create a description table entry for it and specify the repetitions with the descriptive text attribute. (Example: the company logo as one class of instances of the element type “logo”.) (See *6.7.2 Descriptive text*.)

— Any other reuse requirements can be met by using value references (and, if necessary, indirect addresses as provided by the location address module).

NOTE 286 Because extent specifications are architecturally defined as resources (and thus inclusions of the architectural document element), designers can declare event elements in client documents that allow extent or extlist elements as proper subelements or as included subelements.

```
                            <!-- Event -->
<![ %sched; [
<!element
   event          -- Scheduled event --
                  -- Clause: 9.5.1 --
   - O
   (%HyCFC;)*

-- Attributes [base]: overrun --
-- Attributes [sched]: exspec --
-- OptionalAttributes [sched]: pulsemap, objalign --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: modrule:events, prorule:sources --
>
<!attlist
   event          -- Event --
                  -- Clause: 9.5.1 --

   object         -- Event objects --
      CDATA       -- Reference --
      #IMPLIED    -- Default: Event is the object --
>
<!entity % overrun "INCLUDE">
]]><!-- sched -->
```

```
                    <!-- Object alignment rule -->
<![ %objalign; [
<!attlist
-- objalign --    -- Object alignment rule --
                  -- Clause: 9.5.1 --
   (event)

   align          -- Object alignment rule --
                  -- The alignment of an object with an event on each
                     axis can be defined by specifying the name of the
                     axis followed by one of the keywords #LEFT,
                     #RIGHT, or #CENTER.  Alternatively, the name of
                     any axis may be followed by the name of an
                     attribute of the client element whose value
                     determines the position of the object with
                     respect to the event. --
      CDATA       -- Lextype: (AXISNM,(("#LEFT"|"#RIGHT"|#CENTER)|
                                  (ATTNAME,NOTATION)))+ --
                  -- Constraint: Each axis name may appear only
                     once. --
      #IMPLIED    -- Default: no explicit alignment --

-- Attributes named by the object alignment rule attribute have
   an unnormalized lexical type of:

   (s*,granule?,s+,snzi,s+,("STARTS"|"ENDS") #ORDER SGMLCASE,s,char*)

   The first part of the attribute value (up to the last s separator)
   specifies the calibration point within the event.  The rest of the
   value (that which matches char*) is a position within the object,
   and is specified in the notation given following the attribute's
   name in the object alignment attribute. --
>
<!entity % sched "INCLUDE">
]]><!-- objalign -->
```

9.5.2 Event group

The element form **event group** (*evgrp*) is a set of elements that are specified contiguously in a schedule. The elements it contains can be events or other event groups. The significance of these syntactic containers is primarily application-defined. However, HyTime does define certain semantics for the grpdex and grprepet attribute forms (see *9.4.3 Group extent specification*) that can be useful aids in expressing repetitions and derived extents for the events in the content of an event group.

NOTE 287 The grpdex and grprepet attribute forms come in handy, for example, when:

— In a music application, three quarter-notes are given the duration of a half-note (a “triplet”).

— In a slide presentation, the nominal duration of a sequence of slides is compressed to fit the duration of a musical accompaniment.

— When creating a pulsemap, to avoid having to specify repeated pulses individually.

```
                          <!-- Event Group -->
<![ %sched; [
<!element
   evgrp          -- Event group --
                  -- Clause: 9.5.2 --
   - O
   (event|evgrp)+

-- Attributes [base]: overrun --
-- OptionalAttributes [sched]: pulsemap, grpdex, grprepet --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: modrule:events, prorule:sources --
>
]]><!-- sched -->
```

9.6 Objects

Objects are defined by applications, not by HyTime. An object can be represented by any element type, including one with a hierarchical structure.

NOTE 288 It is expected that useful sets of object types will be defined by other standards or by industry groups for public use.

An object could also be a finite coordinate space, or a complete HyTime document. It can be any form of unformatted or formatted character text (including “WYSIWYG”), a structure represented by an SGML element or document, or any other data in any data content notation.

NOTE 289 Some object types that can be useful in multimedia applications are:

still	Still picture: e.g., JPEG image, CAD graphic, fax page, page described by a page description language
audio	Digital audio: e.g., MPEG audio, CD, DAT, Direct-to-disk
midi	Musical instrument control signal sequence
video	Digital video: e.g., MPEG video
applet	Program or script isolated from system: e.g., Java
dance	Choreographic notation: e.g., Labanotation
program	Compiled decision logic: e.g., C++
script	Interpreted decision logic: e.g., interactive hypermedia scripting languages
slide	Slide projector control signals
lighting	Lighting device control signals
studio	Recording studio control signals
device	Analog or digital device control signals
ink	Handwriting: order and pressure of pen strokes

An object could be expressed using parsable character data, an unparseable notation (likely for audio and video), a parsable notation (for example, SPDL for still pictures), or an SGML structure containing a mixture of content types (compound and composite objects). Unparseable notation objects are defined as separate entities, but all entities can be included in a single SDIF data stream for interchange.

9.7 Pulse maps

In some applications, meaningful repetitions, called “pulses”, occur along one or more axes. Any schedule (that is, any evsched, wand, or baton) can serve as a pulse map for any other schedule, or for any event, modscope, proscope, evgrp, modgrp, or progrp.

NOTE 290 For example, a felt beat (“tactus”) in music, a change of frame in video, or a row or column of rectangles in a grid.

Different pulses may have different significance. The significance of a pulse is determined by the application's interpretation of the object of that pulse's event.

NOTE 291 In music, a pulse that represented a stressed beat would have a different object than a pulse that represented an unstressed beat. Since the object of an empty event is considered to be the event itself, different kinds of stresses in music might be represented by different element types, all of which are clients of the *event* element form, or by a single client element type with different values for one of its attributes.

The attribute form **pulse maps** (*pulsemap*) consists of the *pulsemap* attribute, which specifies schedules that establish the pulses intended to provide pulsing contexts for the event, modscope, or proscope that is the referring element or for the events, modscopes, or proscopes scheduled by the referring element.

```
                          <!-- Pulsemaps -->
<![ %pulsemap; [
<!attlist
-- pulsemap --    -- Pulse Maps --
                  -- Clause: 9.7 --
   (baton,event,evgrp,evsched,modgrp,modscope,progrp,proscope,wand)

   pulsemap       -- Pulse Maps --
                  -- Schedules to be used as pulse maps. --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)* --
                  -- Constraint: Pulse maps must be in same FCS as
                     referrer. --
      #IMPLIED    -- Default: none --
>
<!entity % sched "INCLUDE">
]]><!-- pulsemap -->
```

9.8 Dimension referencing

It can be useful to specify all or part of a dimension in terms of components of another dimension. For some applications, this technique can be used to represent alignment and synchronization relationships. This sub-clause specifies HyTime's facilities for dimension references and the constraints imposed on them.

9.8.1 Implicit dimension reference

When the scheduling module is supported, the first marker of a dimension specification in an extent specification of an event, modscope, or proscope can be omitted. The omission of the first marker is an implicit reference to the previous specified event's, modscope's, or proscope's dimension. The omission of the first marker indicates that the “first” component of the dimension with the implicit dimension reference is the next MDU after the highest MDU occupied by the previous specified event, modscope, or proscope on the same axis of the same FCS.

NOTE 292 The effect of this rule is that the implicitly referencing event's first quantum will follow the previous specified event's last quantum on the given axis without a gap.

NOTE 293 This implicit representation of the start of an event allows an extent specification to be portable; that is, it can be referenced for use in other positions, which are determined by the dimension of the event preceding the event whose extent specification contains the reference.

The phrase “previous specified” normally refers to the element that was last parsed before the current element in the SGML representation of the document, in this case, the element of the same HyTime element form (e.g., event) previous to the current one.

NOTE 294 As some objects that occur in events have an inherent extent (for example, audio or video clips), it might seem logical to allow defaulting of the second marker to that value. However, such a rule would require a HyTime engine to understand every possible object notation in order to calculate the extent. Instead, HyTime requires the application that originated the document to specify the inherent extent in a standardized way, such as through a marker function.

9.8.2 Explicit dimension reference

The element form **explicit dimension reference** (*dimref*) references a component of a dimension (the “referenced dimension”) associated with a specified element (the “element specified”). The element specified must be an extent, extlist, event, modscope, proscope, evgrp, modgrp, progrp, evsched, wand, baton or fcs. A set of extents is derived from the element specified, from which the referenced dimension is selected according to parameters supplied as the other attribute values of the dimref. The dimension may be a directly scheduled dimension (that is, directly specified for a given FCS via unprojected extent specifications), or an indirectly scheduled dimension (that is, indirectly specified for a given FCS as a result of projection onto that FCS).

While the syntactic mechanisms used to create dimension references refer to elements, the actual dimensions referenced are those of the events, modscopes and proscopes that occur in the grove constructed by a HyTime engine in the process of interpreting a HyTime document. The dimensions referenced are the dimensions derived from projected and/or unprojected events, modscopes and proscopes after all HyTime processing has been applied, all indirections have been resolved, and all projections have been completed.

Logical sets of events, modscopes, and proscopes that correspond to syntactic scheduling constructs, including fcs elements, evscheds, wands, batons, evgrps, modgrps, and progrps, do not have actual extents. However, for purposes of dimension referencing, such scheduling constructs are regarded as having implicit extents just large enough to contain all of a selected subset of the events, modscopes, and proscopes that they syntactically contain, logically schedule, or logically contain (depending on the selection parameters used in the dimref).

The interpretation and semantics of explicit dimension references are governed by the HyExSpec notation.

In this subclause, the term “directly scheduled” means “scheduled directly in the FCS, rather than being projected there.” The term “projected” means just the opposite: “scheduled indirectly in an FCS by projection.”

The dimref form is described first as it applies to the dimensions of directly scheduled events, modscopes, and proscopes, followed by a description of the additional attributes and constraints for referencing the dimensions of projected events, modscopes, and proscopes.

9.8.2.1 Referencing dimensions of directly scheduled events, modscopes, and/or proscopes

The attribute **element specified** (*elemspec*) refers to the element from which the dimension is to be derived. The specified element must be a client of one of the forms: extent, extlist, event, modscope, proscope, evgrp, modgrp, progrp, evsched, wand, baton, or fcs. It must refer to exactly one element.

NOTE 295 Dimrefs are not references to the syntactic axis markers contained by or referenced from elements. Rather, they are references to dimensions in FCSs resulting from the application of HyTime processing. In order to reference such a resulting dimension, it is necessary to know, at a minimum, the FCS, the axis of that FCS, the extent(s) of the events, modscopes, and proscopes whose dimension(s) are being referenced, and how to interpret those extents. Extents resolve to axis markers, but the granules in which those axis markers are expressed, and which dimensions map onto which axes, cannot be learned from the extents themselves. It is also necessary to know the order in which the markers for the axes are specified in the extents and granules used for each axis. That information comes from the axisord attribute of the governing evsched, wand, or baton.

The element specified is not allowed to be a dimspec because it is not necessarily sensible (a dimspec may appear in a variety of contexts other than an extent of an event, modscope or proscope), it would require more context to be provided (it might not otherwise be clear which axis the dimspec is specified for), and because it is not necessary, due to the fact that even more selectivity is already provided by the extnum and dimnum parameters of the dimref form.

The attribute **selected component of dimension** (*selcomp*) selects the component of the referenced dimension that is to be used: first or lowest quantum (first), last or highest quantum (last), or quantum count (qcnt).

NOTE 296 The number of MDUs per reported quantum is derived from the value of the granule attribute.

The attribute **flip selected component** (*flip*) specifies whether the selected component is to be counted and signed in the usual way, or “flipped”. If the value of the flip attribute is **noflip**, the “first” component is reported as a positive integer corresponding to the number of the first quantum counting from the beginning of the axis, the “last” component is reported as a negative integer corresponding to the number of the last quantum counting from the end of the axis, and the “qcnt” component is reported as a positive integer. If the value of the flip attribute is flip, “first” is reported as a negative integer corresponding to the number of first quantum counting from the end of the axis, “last” is reported as a positive integer corresponding to the number of the last quantum counting from the beginning of the axis, and “qcnt” is reported as a negative integer.

NOTE 297 Flip is useful when performing arithmetic on referenced dimensions. The combinations of selcomp and flip can be summarized informally as follows:

first, noflip	First quantum, counted from beginning (positive).
last, noflip	Last quantum, counted from end (negative).
qcnt, noflip	Size; number of quanta (positive).
first, flip	First quantum, counted from end (negative).
last, flip	Last quantum, counted from beginning (positive).
qcnt, flip	Negative of size (negative).

The attribute **element with the sched attribute form specified** (*schdspec*) specifies exactly one element that has the sched attribute form with values that guide the resolution of the desired referenced dimension. Specifically, the fcs attribute of the sched attribute form specifies the FCS in which the extents associated with the element specified by the elemspec attribute are scheduled, and the axisord attribute specifies both the axis order in which the dimension specifications of the extents are specified, and the granules used to specify the quantum numbers and quantum counts in the dimension specifications.

A value must be specified for the schdspec attribute if the element specified is an extent or extlist. A value may be specified if the element specified is an event, modscope, proscope, evgrp, modgrp, or progrp; in such a case, if no value is specified, the containing element with the sched attribute form is treated as if it were referenced by the value. The value of the schdspec attribute is ignored if the element specified is an evsched, wand, baton, or fcs.

The element with the sched attribute form specified by the schdspec attribute must be one that governs the scheduling of the extents associated with the element specified by the elemspec attribute.

NOTE 298 Any given event, modscope, proscope, evgrp, modgrp, or progrp element may appear in more than one evsched, wand, or baton element. For example, an evsched may incorporate an event by means of a HyTime valueref attribute. In such a case, if the referenced dimension is governed by the incorporating evsched, the incorporating evsched should be specified explicitly by the schdspec attribute.

Extents and extlists are resources that can appear syntactically outside the context of any element with the sched attribute form; this is why, when the element specified is an extent or extlist, a value must be specified for the schdspec attribute.

NOTE 299 In cases where the element specified is evsched, wand, or baton, the value of the schdspec attribute is ignored because these elements themselves use the sched attribute form. In cases where the element specified is an fcs, it is ignored because there is no need to interpret specific extent specifications in specific scheduling contexts.

The attribute **axis specified** (*axisspec*) specifies the axis on which the referenced dimension occurs. A value must be specified unless the schdspec attribute specifies or defaults to an evsched, baton, or wand whose sched attribute form specifies only one axis; if there is only one axis, the value of the axisspec attribute is ignored.

The attribute **dimension number** (*dimnum*) selects the referenced dimension from a set of dimensions.

NOTE 300 For example, if no value is specified for the extnum attribute, and there is more than one extent, dimnum can be used to select a single dimension.

The value of the dimnum attribute is the number of the corresponding dimension; a value of 1 (one) indicates the first dimension. The order of the dimensions is not determined by their lexical order of appearance in the document. Instead, the dimensions are sorted primarily by the number of the first quantum (counted starting with the first quantum on the axis), and secondarily by the qcnt, where lower quantum numbers come before higher ones, and smaller qcnts come before larger ones. The order of equivalent dimensions is undefined.

If there is more than one dimension and there is no dimnum specification, the referenced dimension is the smallest dimension that would encompass all the dimensions.

When the element specified is an event, modscope, or proscope that is scheduled with multiple extents, or when multiple extents are associated with the element specified by virtue of its containment of multiple events, modscopes, or proscopes, or both, the attribute **extent number** (*extnum*) specifies which syntactic extent to use in determining the referenced dimension. The extnum attribute is ignored if the element specified is an extent.

NOTE 301 By definition, an extent element defines only a single extent.

The use of notations (such as the HyExSpec notation) and marker functions makes it possible to describe lists of extents (extlists) in a single element whose content does not mark up each extent specification as an individual extent element. In such a case, the element is regarded as a single extent for purposes of extnum specifications.

NOTE 302 In such a case, the value of the dimnum attribute can be used to select individual dimensions, if desired.

An extnum value of 1 (one) indicates the first extent. The order of the extents is determined first by the syntactic order of the selected elements whose exspec attributes specify the extents, secondarily by the syntactic order of their specifications in the values of the relevant exspec attributes, and finally by the order of appearance of the extents in the source document (that is, within the specified extlists). If there is more than one extent and no value is specified for the extnum attribute (or if the selected extent has more than one dimension), and there is no dimnum specification, the referenced dimension is the smallest dimension that would encompass all the dimensions.

NOTE 303 Only events, modscopes and proscopes have exspec attributes. If, for example, the element specified by the elemspec attribute is an evsched containing two evgrps, and each evgrp contains three events, and each event has one extent, then there are six extents that can be selected using a value for the extnum attribute of 1, 2, 3, 4, 5, or 6. If no value is specified for the extnum attribute, all six extents are selected. (If no value is specified for the dimnum attribute, either, the referenced dimension is a fictitious one that is just big enough to contain all six.)

NOTE 304 Extnum selects from syntactic constructs in an order derived from the syntax of the document. Dimnum is entirely different; it selects from logical dimensions in the order in which they appear on the axes of fully processed logical FCSs.

When the element specified is an evsched, wand, baton, or fcs, the attribute **directly scheduled types for calculating referenced dimension** (*dsdtypes*) limits the types of constructs that will be taken into account when calculating the referenced dimension. The value of the dsdtypes attribute is a list of tokens, either keywords corresponding to the various scheduling forms or the GIs of client elements derived from scheduling forms.

The keyword #ALL causes all construct types to be taken into account. If #ALL is not specified, only the construct types that are specified in the value will be taken into account. All types of events, modscopes, proscopes, evgrps, modgrps, progrps, evscheds, wands, batons, and FCSs will be taken into account if the value contains keywords corresponding to those forms: #EVENT, #MODSCOP, #PROSCOP, #EVGRP, #MODGRP, #PROGRP, #EVSCHED, #WAND, #BATON, or #FCS. In addition, all elements whose GIs appear as tokens in the value of the dsdtypes attribute, and that are events, modscopes, proscopes, evgrps, modgrps, progrps, evscheds, wands, batons, or fcs elements, will be taken into account.

NOTE 305 It would be redundant to specify, for example, both #PROSCOP and the GI of a proscope element in the value of an dsdtypes attribute, although it is not an error to do so.

Any combination of the foregoing keywords and GIs may be used in the value of the dsdtypes attribute. If no value is specified, the default value will depend on the form of the element specified, according to the following table:

extent	(null string)
extlist, event, modscope, or proscope	“#EXTENT #EXTLIST”
evgrp or evsched	“#EXTENT #EXTLIST #EVENT #EVGRP”
modgrp or wand	“#EXTENT #EXTLIST #MODSCOP #MODGRP”
progrp or baton	“#EXTENT #EXTLIST #PROSCOP #PROGRP”
fcs	“#ALL”

The attribute **granule for reporting** (*granule*) names the granule to be regarded as the quantum used to report the referenced dimension. If no value is specified, the referenced dimension is reported in terms of the MDU in effect on the axis of the referenced dimension. The granule specified must be defined in terms of the same SMU as the axis specified by the axisspec attribute. If the granule is larger than the MDU of an axis, there may be less than one granule's worth of MDUs at the end of the axis after the last granule.

NOTE 306 In other words, the last granule's last MDU may not be the last MDU on the axis, with the number of remaining MDUs equal to the remainder after dividing the number of MDUs on the axis by the number of MDUs per granule.

The attribute **round off to granule handling** (*roffgran*) specifies how to handle a situation in which reporting the referenced dimension using the granule specified by the granule attribute would not be exactly accurate. The value **round off error** (*roerr*) indicates that if rounding is required, it is an error. The value **round off to minimum complete dimension** (*rocmplet*) indicates that the minimum dimension expressible in terms of the granule specified that completely covers the actual dimension will be reported. The value **round off to maximum complete dimension** (*roscant*) indicates that the maximum dimension expressible in terms of the granule specified that is fully occupied by the actual dimension will be reported. The value **round off to dimension**

boundaries (*roboundy*) indicates that the dimension that places the boundaries of the reported dimension as close as possible to the boundaries of the actual dimension is reported.

NOTE 307 The use of recursive dimension references and selected components, in conjunction with marker functions, allows the specification of arbitrarily complex synchronization and alignment relationships. These tools allow applications to place and size events, modscopes, and proscopes in relation to other events, modscopes, and proscopes.

```
                <!-- Explicit Dimension Reference -->
<![ %dimref; [
<!notation
   dimref         -- Explicit dimension reference marker function
                     notation --
                  -- Clause: 9.8.2.1 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Explicit dimension reference marker function notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!element
   dimref         -- Explicit dimension reference --
                  -- Clause: 9.8.2.1 --
   - O
   (%HyCFC;)*

-- Attributes [sched]: dimref --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dimref         -- Explicit dimension reference --
                  -- Clause: 9.8.2.1 --

   HyBase   NAME     #FIXED markfun
   notation NOTATION (dimref) #FIXED dimref

   elemspec       -- Element specified --
                  -- Element from which referenced dimension is
                     derived --
      CDATA       -- Reference --
                  -- Reftype: (baton|event|evgrp|evsched|extent|
                               extlist|fcs|modgrp|modscope|progrp|
                               proscope|wand) --
      #REQUIRED

   selcomp        -- Selected component of dimension --
                  -- Dimension component of specified element to be
                     referenced.
                        first:    First or lowest quantum
                        last:     Last or highest quantum
                        qcnt:     Quantum count --
      (first|last|qcnt)
      qcnt

   flip           -- Flip selected component --
                  -- Whether or not to invert the referenced dimension
                     component.  Specifically:
                        noflip:   First is reported as a positive
                                  integer corresponding to the number
                                  of the first quantum counting from
                                  the beginning of the axis, last is
                                  reported as a negative integer
                                  corresponding to the number of the
                                  last quantum counting from the end of
                                  the axis, and qcnt is reported as a
                                  positive integer.
                        flip:     First is reported as a negative
                                  integer corresponding to the number
                                  of first quantum counting from the
                                  end of the axis, last is reported as
                                  a positive integer corresponding to
                                  the number of the last quantum
                                  counting from the beginning of the
                                  axis, and qcnt is reported as a
                                  negative integer. --
      (flip|noflip)
      noflip

   schdspec       -- Element with sched attribute form specified --
                  -- Establishes context in which extents are
                     interpreted: FCS, axis order, and HMU --
                  -- Constraint: Ignored if element specified is
                     evsched, wand, baton or fcs. --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)? --
      #IMPLIED    -- Default: dimref is treated as if the containing
                     element with the sched attribute form is
                     referenced by schdspec. --
                  -- Constraint: Required if elemspec is extent or
                     extlist. --

   axisspec       -- Axis specified --
                  -- Axis on which referenced dimension occurs --
      NAME        -- Lextype: AXISNM --
                  -- Constraint: must be valid axis name of FCS in
                     which the referenced dimension occurs. --
      #IMPLIED    -- Default: only axis --

   dimnum         -- Dimension number --
      NUMBER      -- Constraint: 1 = first dimension in order of
                     appearance on axis --
      #IMPLIED    -- Default: smallest dimension encompassing all
                     dimensions --

   extnum         -- Extent number --
      NUMBER      -- Constraint: 1 = first or only extent in order of
                     specification/appearance in document --
      #IMPLIED    -- Default: combination of all relevant extents --
```

```
granule        -- Granule for reporting --
               -- Granule for reporting referenced dimension --
   CDATA       -- Lextype: granule --
               -- Constraint: granule must be defined in terms of
                  the same SMU as axis --
   #IMPLIED    -- Default: MDU of axis of referenced dimension --

roffgran       -- Round off to granule handling --
               -- Specifies how to handle roundoff.
                     roerr:    Report error if rounding is required.
                     rocmplet: Round off to minimum complete
                               dimension; that is, minimum dimension
                               expressible in terms of the granule
                               specified that completely covers the
                               actual dimension.
                     roscant:  Round off to maximum complete
                               dimension; that is, maximum dimension
                               expressible in terms of the granule
                               specified that is fully occupied by
                               the actual dimension
                     roboundy: Round off to dimension boundaries;
                               that is, dimension that places the
                               boundaries of the reported dimension
                               as close as possible to the
                               boundaries of the actual
                               dimension. --
   (roboundy|rocmplet|roerr|roscant)
   roboundy

dsdtypes       -- Directly scheduled types for calculating
                  referenced dimension --
               -- Types of directly scheduled elements to be taken
                  into account when calculating the referenced
                  dimension. --
   CDATA       -- Lextype: ("#ALL"|("#BATON"|"#EVENT"|"#EVGRP"|
                                  "#EVSCHED"|"#EXTENT"|
                                  "#EXTLIST"|"#FCS"|"#MODGRP"|
                                  "#MODSCOP"|"#PROGRP"|
                                  "#PROSCOP"|"#WAND"|GI)*) --
               -- Constraint: GIs must be of event, modscope,
                  proscope, evgrp, modgrp, progrp, evsched, wand,
                  baton, or fcs form elements. --
   #IMPLIED    -- Default: Depends on form of element specified, as
                  follows:
                   element specified:    default value:
                   - - - - - - - - - -   - - - - - - - -
                   extent                (null string)
                   extlist, event,
                   modscope or proscope  "#EXTENT #EXTLIST"
                   evgrp or evsched      "#EXTENT #EXTLIST #EVENT
                                          #EVGRP"
                   modgrp or wand        "#EXTENT #EXTLIST #MODSCOP
                                          #MODGRP"
                   progrp or baton       "#EXTENT #EXTLIST #PROSCOP
                                          #PROGRP"
```

```
                        fcs                   "#ALL" --
>
<!entity % sched "INCLUDE">
]]><!-- dimref -->
```

9.8.2.2 Referencing dimensions of indirectly scheduled events, modscopes, and/or proscopes

When the dimension referenced is derived from the dimensions of one or more indirectly scheduled (that is, projected) events, modscopes or proscopes, additional contextual information may be needed and additional constraints apply. The extents of the set of events, modscopes, and proscopes whose projected dimensions are being referenced are selected in the same way as if their directly scheduled dimensions were being referenced, using the elemspec, schdspec, extnum, and dsdtypes attributes. Also, the reporting of the referenced dimension is governed by the granule, roffgran, selcomp, and flip attributes in the same way as a directly scheduled dimension. However, the dimnum and axisspec attributes refer to the projected dimension number and the target axis, respectively, instead of the unprojected dimension number and the axis on which it was directly scheduled.

When the element specified is an evsched, wand, baton, or fcs, the attribute **projected and/or direct** (*prjdirct*) indicates whether the set of events, modscopes, and proscopes from which the referenced dimension is derived includes the events, modscopes, and proscopes that are directly scheduled in or by the element specified (drctonly), or only the dimensions of events, modscopes and proscopes that are projected on the element specified (projonly), or both (drctproj). The value of the prjdirct attribute is ignored if the element specified is an extent, extlist, event, modscope, proscope, evgrp, modgrp or progrp. If a value is specified for the extnum attribute, the value of the prjdirct attribute will be ignored, and the dimref will be treated as if the value of the prjdirct attribute were specified to be drctonly.

NOTE 308 Because extnum selects the unprojected referenced dimension according to its syntactic specification in the source document, extnum cannot be used to select extents that have been projected onto an fcs, evsched, wand, or baton that is the element specified by the elemspec attribute. The dimensions of such extents can be referenced by means of the dimnum attribute, in their order of appearance on an axis. Alternatively, they could be referenced by using the elemspec attribute to specify an element that would permit the selection of their unprojected extents.

NOTE 309 If the element specified is an evsched, wand, baton, or fcs, any events, modscopes, or proscopes that exist in it as a result of having been projected onto it will not be taken into account unless all of the kinds of elements that play a role in projecting such events, modscopes, or proscopes are specified in the value of the dsdtypes attribute. Specifying “#ALL” assures that all events, modscopes, and proscopes projected onto the element specified will be taken into account, regardless of which kinds of elements played a role in projecting them.

The attribute **projected to target** (*prjtarg*) specifies an element onto which the selected extent (or all of the specified extents if no value is specified for the extnum attribute) of the unprojected events, modscopes, and proscopes may have been projected. The prjtarg attribute may specify only one element, which must be an fcs, evsched, wand, or baton. If the element specified by the elemspec attribute is an extent or extlist, the value of the prjtarg attribute is ignored. If a value is specified for the prjtarg attribute and that value is not ignored:

— The referenced dimension is the dimension of the selected extent (or all of the extents if no value is specified for the extnum attribute) of the unprojected events, modscopes, and proscopes selected for calculation of the unprojected referenced dimension, as projected onto the fcs, evsched, wand or baton specified by the prjtarg attribute, and subject to further selection by the prjby attribute (see below).

— The value of the axisspec attribute must be a valid axis name of the target FCS. The referenced dimension occurs on that axis.

— The value of the dimnum attribute does not select from the unprojected referenced dimensions; instead, it selects from the projected referenced dimensions.

The attribute **projected by** (*prjby*) limits the projections of the selected extent (or all of the extents if no value is specified for the extnum attribute) of the unprojected events, modscopes, and proscopes selected for calculation of the unprojected referenced dimension, as projected onto the fcs, evsched, wand or baton specified by the prjtarg attribute, that will be taken into account, to only those projections that are specified by the prorule, baton, progrp and proscope elements specified by the prjby attribute. If no value is specified for the prjby attribute, all projections specified by all relevant prorules, batons, progrps, and proscopes will be taken into account.

It is a reportable HyTime error if a non-ignored value is specified for the prjtarg attribute and the projection targets include more than one FCS.

NOTE 310 The value of the prjby attribute should be used in such a way as to avoid errors of this kind.

If no value is specified for the prjtarg attribute, or if the value of the prjtarg attribute is ignored, the value of the prjby attribute is ignored.

```
                <!-- Projected Dimension Reference -->
<![ %dimref; %project; [
<!attlist
-- pdimref --      -- Projected dimension reference --
                   -- Clause: 9.8.2.2 --
                   -- Dimref attributes for references to projected
                      sets of events, modscopes and/or proscopes --
   (dimref)

   prjdirct        -- Projected and/or direct --
                   -- Selection on the basis of whether events,
                      modscopes, or proscopes were either projected
                      onto, or directly scheduled in, the specified
                      element.
                         drctonly: Direct only
                         projonly: Projected only
                         drctproj: Direct and projected --
                   -- Constraint: Value is ignored if element specified
                      is not evsched, wand, baton, or fcs. --
                   -- Constraint: Value is ignored (is, in effect,
                      drctonly) if a value is specified for extnum. --
      (drctonly|drctproj|projonly)
      drctonly

   prjtarg         -- Projected to target --
                   -- Referenced dimension exists by projection onto
                      element specified by prjtarg attribute. --
                   -- Constraint: Value is ignored (and referenced
                      dimension is not a projected dimension) if
                      element specified is extent or extlist. --
      CDATA        -- Reference --
                   -- Reftype: (baton|evsched|fcs|wand) --
      #IMPLIED     -- Default: Referenced dimension is unprojected. --

   prjby           -- Projected by --
                   -- Referenced dimension exists by projection onto
                      element specified by prjtarg attribute only by
                      elements specified by prjby attribute. --
                   -- Constraint: Value is ignored unless prjtarg has
                      non-ignored value. --
      CDATA        -- Reference --
                   -- Reftype: (baton|progrp|prorule|proscope) --
      #IMPLIED     -- Default: All relevant prorules, batons, progrps,
                      and proscopes will be taken into account. --
>
]]><!-- dimref, project -->
```

9.9 Calibrated real time axes

Any axis of a finite coordinate space can be calibrated with respect to real time. This International Standard provides a notation for specifying real time calibration values, HyTime Calendar Specification Notation (HyCalSpc). It also defines a specialized marker function element, calendar specification (calspec), for use with axes calibrated to real time using the HyCalSpc notation.

9.9.1 HyTime calendar specification notation

The notation **HyTime Calendar Specification Notation** (*HyCalSpc*) is used to specify a point in past or future real time in terms of Julian dates or ordinary calendar dates and times of day, with optional adjustments for daylight saving and time zones. The point specified is that occurring at the beginning of the smallest unit of time in the terms by which the point is specified (the smallest granularity available is a Systeme International second).

NOTE 311 For example, the calendar specification “1996-02-07” identifies the point occurring at midnight between February 6th and February 7th of 1996.

The HyCalSpc notation can be used to calibrate axes measured in real time. The *calibrate* attribute of the fcs element is used to specify the notation in which an axis is calibrated (HyCalSpc), and to give the name of another attribute of the fcs element whose value gives the external context's calibration point, expressed in HyCalSpc notation (see *9.3.1 Axis calibration*). Although HyCalSpc notation allows the year not to be specified, it is an error to calibrate an axis without a specification that conforms to either the Julian date or fulldate lexical type.

The HyCalSpc notation also forms the basis of the HyCalFun marker function notation; see *9.9.2 Calendar specification*.

The **GMT or UTC timescale** (*GMTorUTC*) attribute specifies whether the time expressed is in terms of Greenwich mean time (GMT) or Coordinated Universal Time (UTC). UTC is the default.

NOTE 312 The HyCalSpc notation is described in terms of lexical type declarations, some of which are expressed using Posix regular expressions (REGEX notation) as well as HyLex expressions. The regex notation is declared only as the grammar used to describe the HyCalSpc notation. It is not necessary for applications to support regex notation in order to support HyCalSpc notation; it is only necessary that they be capable of supporting the HyCalSpc notation itself (see *C.1 HyTime Lexical Types*).

```
           <!-- HyTime Calendar Specification Notation -->
<![ %HyCalSpc; [
<!notation
   HyCalSpc       -- HyTime Calendar Specification Notation --
                  -- Clause: 9.9.1 --
                  -- Specifies a point in real time. The point
                     specified is that which occurs at the beginning
                     of the smallest unit of time in terms of which
                     the point is specified. --
                  -- Note: For example, the calendar specification
                     "1996-02-07" identifies the point occurring at
                     midnight between February 1st and February 2nd of
                     1996. --
                  -- Note: Can be used directly for axis calibration
                     and alignment.  Also used as the basis for a
                     marker function notation; see HyCalFun. --
                  -- Lextype: ((JULDATE|fulldate|monthday|day)?,
                               (UTCtime|hrminsec|hrmin|minute)?,
                               timeoff*) --
                    -- Constraint: Date must be valid (e.g. not
                       "2-30"). --
                    -- Constraint: Valid time followed by list of
                       applicable non-conflicting time offsets for
                       daylight-saving or time zones. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Calendar Specification Notation//EN"

-- Attributes [sched]: HyCalSpc --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyCalSpc         -- HyTime Calendar Specification Notation --
                    -- Clause: 9.9.1 --

   GMTorUTC         -- GMT or UTC timescale --
                    -- Note: (GMT noon = UTC midnight) --
      (GMT|UTC)
      UTC
>
<!entity % sched "INCLUDE">
]]><!-- HyCalSpc -->
```

9.9.2 Calendar specification

The **calendar specification** (*calspec*) element form is based on the marker function (markfun) form (see *6.8.1.2 Marker Functions*). It is used in the context of axis marker list notation data (see *6.8.1 HyTime axis marker list notation*) to specify a single marker in terms of the corresponding calendar date and time. Calendar specifications can be used to express dimensions only on axes calibrated using the HyCalSpc notation.

The content of calendar specifications is always expressed in **HyTime calendar marker function notation** (*HyCalFun*), which is exactly like HyCalSpc notation (see *9.9.1 HyTime calendar specification notation*) except that it returns an axis marker. If a calendar specification does not specify a year (by specifying either a Julian date or a fulldate), the year specified or implied by the previously specified (that is, syntactically previous) calspec is assumed. If there is no previously-specified year, the year is taken to be the first year of the axis.

If a calspec is used as the first marker of a dimension specification, a positive marker is returned which is the number of the quantum corresponding to the date and time specified in its content, counting from the beginning of the axis. If the calspec is used as the second marker of a dimension specification, a negative marker is returned, giving the number of the quantum corresponding to the date and time specified in its content, counting backwards from the end of the axis.

```
                     <!-- Calendar Specification -->
<![ %calspec; [
<!notation
   HyCalFun         -- HyTime Calendar Marker Function Notation --
                    -- Clause: 9.9.2 --
                    -- A calendar specification is a marker function to
                       be used to specify a quantum along a coordinate
                       axis whose measurement domain is real time (that
                       is, SIsecond) and that has been calibrated using
                       HyCalSpc notation.  If used as the first marker
                       of a dimension specification, a positive marker
                       is returned.  If used as the second marker of a
                       dimension specification, a negative marker is
                       returned. --
                    -- Constraint: highest-order date/time component
                       specified cannot exceed that specified for axis
                       calibration. --
                    -- Constraint: omitted date/time components are
                       those of previous specified calspec for this
                       schedule. --
                    -- Lextype: ((JULDATE|fulldate|monthday|day)?,
                                 (UTCtime|hrminsec|hrmin|minute)?,
                                 timeoff*) --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Calendar Marker Function Notation//EN"

-- Attributes [sched]: HyCalFun --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyCalFun       -- HyTime Calendar Marker Function Notation --
                    -- Clause: 9.9.2 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyCalSpc

   GMTorUTC       -- GMT or UTC timescale --
                    -- Note: (GMT noon = UTC midnight) --
        (GMT|UTC)
        UTC
>
<!element
   calspec        -- Calendar Specification --
                    -- Clause: 9.9.2 --
   O O
   (%HyCFC;)*     -- Lextype: ((JULDATE|fulldate|monthday|day)?,
                                 (UTCtime|hrminsec|hrmin|minute)?,
                                 timeoff*) --
                    -- Constraint: If neither JULDATE nor fulldate is
                       present, previously specified year and other
                       larger quanta will be assumed. --

-- Attributes [sched]: calspec --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   calspec        -- Calendar Specification --
                    -- Clause: 9.9.2 --
   HyBase   NAME     #FIXED markfun
   notation NOTATION (HyCalFun) #FIXED HyCalFun

   GMTorUTC       -- GMT or UTC timescale --
                  -- Note: (GMT noon = UTC midnight) --
      (GMT|UTC)
      UTC
>
<!entity % HyCalSpc "INCLUDE">
]]><!-- calspec -->
```

9.10 Finite coordinate space location address

The element form **FCS location address** (*fcsloc*) addresses events, modscopes, and proscopes in finite coordinate spaces by specifying the extent of the "selection boundary" within the FCS that contains the events to be addressed. The fcsloc addresses ("selects") a set of events, modscopes, and/or proscopes, or a set of the objects, modifiers, and/or projectors scheduled by them.

The location source of an fcsloc must be one or more fcs, evsched, wand, or baton elements. There is no default location source: the location source attribute must be specified. If the location source consists of multiple objects, the FCSs with which they are associated must all have the same number of axes.

The content of an fcsloc is the extent specification that defines the extent of the selection boundary. Multiple extents are allowed. Dimensions are expressed in terms of the HMUs specified by the axis attributes of all schedule elements serving as location sources, or in terms of MDUs if the location source is an FCS.

The attribute **precision of selection** (*select*) defines the precision with which the objects are selected by specifying how much of each event must be contained by the boundary of the fcsloc. For each axis in the FCS the attribute specifies whether the event must be entirely within the selection boundary (#ALL), the minimum number of quanta that must be within the selection boundary (*number*), or that the event is selected if it has even one quantum within the selection boundary (#ANY). The value is either one for each axis, in the order specified by the axes attribute (if the location source is an fcs) or the axisord attribute (if the location source is a schedule), or one for all axes.

The attribute **event or objects?** (*objects*) specifies whether the fcsloc addresses the events, modscopes, and/or proscopes selected ("events", which is the default) or the objects, modifiers, and/or projectors scheduled by them ("objects"). When the value of the objects attribute is "objects", the fcsloc is equivalent to the location ladder formed by making the fcsloc the location source for a property location that addresses the "objects" property of the events, modscopes and proscopes selected by the fcsloc.

The attribute **directly scheduled types to select** (*dsdtypes*) limits the types of constructs that are selected. The value of the dsdtypes attribute is a list of tokens, including keywords corresponding to the event ("#EVENT"), modscope ("#MODSCOP") and proscope ("#PROSCOP") element forms, and the GIs of client elements of those forms. The keyword #ALL causes all events, modscopes, and proscopes to be selected. If #ALL is not specified, only the client element types that are specified in the value will be selected. All types of events, modscopes and proscopes will be selected if the value contains the keywords corresponding to those forms. In addition, all elements whose GIs appear as tokens in the value of the dsdtypes attribute, and that are events, modscopes or proscopes, will be selected.

NOTE 313 It would be redundant to specify, for example, both #PROSCOP and the GI of a proscope element in the value of an dsdtypes attribute, although it is not an error to do so.

Any combination of the foregoing keywords and GIs may be used in the value of the dsdtypes attribute. If no value is specified, the default value will depend on the form of the element specified, according to the following table:

evsched	"#EVENT"

wand	“#MODSCOP”
baton	“#PROSCOP”
fcs	“#ALL”

If the rendition module is supported, three additional attributes may be used to establish further selection criteria.

The attribute **projected and/or direct** (*prjdirct*) indicates whether the set of events, modscopes, and proscopes from which the fcsloc selects includes only the events, modscopes, and proscopes that are directly scheduled in or by the locsrc elements (drctonly), or only the events, modscopes and proscopes that are projected on the locsrc elements, (projonly), or both (drctproj).

The attribute **projection sources** (*prjsrc*) is used to limit the projected events, proscopes and modscopes selected to those that are directly scheduled in or by the specified event, modscope, proscope, evgrp, modgrp, progrp, evsched, wand, baton, or fcs client elements. The value is ignored if the value of the prjdrct attribute is drctonly. If no value is specified, the selection of projected events, modscopes, and proscopes is not limited with respect to their sources.

The attribute **projected by** (*prjby*) is used to limit the projected events, proscopes and modscopes selected to those that are projected by means of the prorule, baton, progrp, and proscope client elements specified in its value. The value is ignored if the value of the prjdrct attribute is drctonly. If no value is specified, the selection of projected events, modscopes, and proscopes is not limited with respect to the elements used to specify their projection to the locsrc elements.

```
          <!-- Finite coordinate space location address -->
<![ %fcsloc; [
<![ %HyExtLst; [
   <!entity % dfcslcnt "HyExtLst">
]]>
<!entity %
   dfcslcnt       -- Default fcsloc extent list notation --
                  -- Clause: 9.10 --

   "#REQUIRED"
>
<!element
   fcsloc         -- Finite coordinate space location address --
                  -- Clause: 7.10.2 --
                  -- Addresses events, modscopes, and proscopes (or
                     the objects, modifiers, and projectors they
                     contain) in finite coordinate spaces. --
                  -- Constraint: location source must be an FCS, event
                     schedule, wand, or baton --
   - O
   (%dimlist;|extent|extlist)*
                  -- Constraint: if no extent list notation is
                     specified, content must consist of only extent
                     and extlist elements. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- Attributes [sched]: fcsloc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- OptionalAttributes [sched]: rfcsloc --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   fcsloc         -- Finite coordinate space location address --
                  -- Clause: 9.10 --

   HyBase   NAME     #FIXED queryloc

   notation       -- Extent list notation for selection --
      NAME        -- Lextype: NOTATION --
      %dfcslcnt;  -- Default: content must consist of only extent and
                     extlist elements. --

   select         -- Precision of selection --
                  -- Description: How much of an event, modscope, or
                     proscope must appear within the selection
                     boundary in order for it (or the object,
                     modifier, or proscope that it schedules) to be
                     selected. --
      CDATA       -- Lextype: ("#ALL"|"#ANY"|(unzi, granule?))+ --
                  -- Constraint: one for each axis of location source
                     or one for all axes. --
                  -- Constraint: Precision unzi is specified in terms
                     of the following granule, if supplied; otherwise
                     in terms of the axis HMU defined for the schedule. --
      "#ALL"      -- Default: event, modscope, or proscope must be
                     entirely within selection boundary in order to be
                     selected. --

   objects        -- Events or objects? --
                  -- Node type to select: events, modscopes, or
                     proscopes (events) or the objects, modifiers, and
                     projectors that are scheduled by them
                     (objects)? --
      (events|objects)
      events

   dsdtypes       -- Directly scheduled types to select --
                  -- Types of directly scheduled elements to be
                     selected. --
      CDATA       -- Lextype: ("#ALL"|("#EVENT"|"#MODSCOP"|
                                       "#PROSCOP"|GI)*) --
                  -- Constraint: Can be any combination of #EVENT,
                     #MODSCOP, #PROSCOP, and the GIs of event,
                     modscope or proscope form elements.  Can also be
                     #ALL. --
      #IMPLIED    -- Default: For each form of element used as locsrc:
                       locsrc element:       default value:
                       - - - - - - -         - - - - - - -
                         evsched               "#EVENT"
                         wand                  "#MODSCOP"
                         baton                 "#PROSCOP"
                         fcs                   "#ALL"
                     --
```

```
>
<!entity % sched "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- fcsloc -->

<![ %fcsloc; %project; [
<!attlist
-- rfcsloc --     -- Finite coordinate space location address with
                     rendition module --
                  -- Clause: 7.10.2 --
   (fcsloc)

   prjdirct       -- Projected and/or direct --
                  -- Selection based on how objects were scheduled:
                      drctonly: (Direct only) Do not select the
                                events, modscopes, and proscopes
                                projected onto the element specified.
                      projonly: (Projected only) Do not select the
                                events, modscopes, and proscopes
                                directly scheduled in or by the
                                element specified.
                      drctproj: (Direct and projected) Do not exclude
                                direct or projected events, modscopes,
                                and proscopes from being selected. --
      (drctonly|drctproj|projonly)
      drctonly

   prjsrc         -- Projection sources --
                  -- Of the events, modscopes and proscopes indirectly
                     scheduled on the locsrc elements, select only
                     those that are directly scheduled by or in the
                     specified elements. --
      CDATA       -- Reference --
                  -- Reftype: (baton|event|evgrp|evsched|fcs|modgrp|
                               modscope|progrp|proscope|wand)* --
                  -- Constraint: Value is ignored if value of prjdrct
                     is drctonly. --
      #IMPLIED    -- Default: No projection sources constraint. --

   prjby          -- Projected by --
                  -- Of the events, modscopes and proscopes indirectly
                     scheduled on the locsrc elements, select only
                     those that are projected there by the specified
                     projection elements. --
      CDATA       -- Reference --
                  -- Reftype: (baton|progrp|prorule|proscope)* --
                  -- Constraint: Value is ignored if value of prjdrct
                     is drctonly. --
      #IMPLIED    -- Default: No projection elements constraint. --
>
]]><!-- fcsloc, project -->
```

10 Rendition module

This clause describes the optional rendition module of HyTime, which includes both the modification facilities and the projection facilities. The module requires support of the scheduling module, and either or both of the “modify” and “project” options of the rendition module must be supported.

The modification and projection facilities of HyTime allow some of the parameters of an application's rendition of a document to be specified in a standardized way. Modification, which is described in *10.2 Object Modification*, is used to specify processing of information objects. Projection, which is described in *10.3 Projection*, is used to specify the semantic copying of events, modifier scopes, and projector scopes from schedule to schedule, perhaps transforming their positions and extents in the process. Events and modscopes can be projected with their objects and modifiers already modified in their source contexts, or with their objects and modifiers in unmodified form. A complete rendition can include several sets of modifications and/or projections; a container form for expressing complete renditions is described in *10.4 Rendition rule*.

10.1 Common rendition attributes

The following attribute forms are used by both modification and projection rendition forms.

10.1.1 Precision of Selection

When used with baton, progrp, and proscope, the attribute **precision of selection** (*select*) specifies the precision with which events, modscopes, and proscopes to be projected are selected. When used with wand, modgrp, and modscope, the attribute specifies the precision with which events and modscopes (whose objects and modifiers are to be modified) are selected. The attribute specifies how much of each event, modscope or proscope must be contained within the extent of each modifying modscope or projecting proscope in order to qualify for the application of object modification or projection.

NOTE 314 Projectors can be applied to events, modscopes, and proscopes. Modifiers can be applied to information objects and to other modifiers, but not to projectors.

For each axis in the FCS, the *select* attribute specifies that the event, modscope, or proscope that is to be projected or the event or modscope whose object or modifier is to be modified must be entirely within the extent of the modifying modscope or projecting proscope (#ALL) in order to be selected, that it must have a minimum number of quanta within the extent of the modifying modscope or projecting proscope (*number*) in order to be selected, or that it is selected if it has even one MDU within the extent of the modifying modscope or projecting proscope (#ANY). Modification or projection occurs if and only if the minimum requirement specified for all axes is satisfied.

For a wand, if no value is specified, only the objects and modifiers of events and modscopes that are entirely within the extent of the modscopes scheduled by the wand will be modified (#ALL). For a modgrp or modscope, the default value is the same as the effective value of the nearest containing modgrp or wand.

For a baton, if no value is specified, only events, modscopes and proscopes that are entirely within the extent of the proscopes scheduled by the baton will be projected (#ALL). For a progrp or proscope, the default value is the same as the effective value of the nearest containing progrp or baton.

The attribute **portion affected** (*portion*) specifies whether a modifying modscope or projecting proscope applies to the entire to-be-modified or to-be-projected event, modscope or proscope (“whole”) or only to the part that is co-located with the extent of the modifying modscope or projecting proscope (“part”). A single value can be used for all axes, or a separate value can be used for each axis.

For a wand, if no value is specified, the default for all modscopes in the wand is “whole”. For a modifying modgrp or modscope, the default value is the same as the effective value of the nearest containing modgrp or wand. For a baton, if no value is specified, the default for all proscopes in the baton is “whole”. For a progrp or proscope, the default value is the same as the effective value of the nearest containing progrp or baton.

NOTE 315 In the case of object modification, the application of the effect of the portion attribute is beyond the realm of pure HyTime processing, in the same way and for the same reasons that pure HyTime processing does not normally concern itself with the interpretation of objects or modifiers. The portion attribute is provided merely to allow authors to state whether or not the extent of a modscope is intended to act as a limiting parameter of the object modification process, once an event or modscope has already been selected for object modification. If the value is “whole” for all axes, the extent of the modifying modscope is not intended to be used as such a parameter.

```
                 <!-- Precision of selection -->
<![ %rend; [
<!attlist
-- select --      -- Precision of selection --
                  -- Clause: 10.1.1 --
   (baton,modgrp,modscope,progrp,proscope,wand)

   select         -- Precision of selection --
                  -- How much of the event, modscope or proscope must
                     lie within the extent of the modifying modscope
                     or projecting proscope in order for modification
                     or projection to occur. --
      CDATA       -- Lextype: ("#ALL"|"#ANY"|(unzi, granule?))+ --
                  -- Constraint: One specification for each axis or
                     one for all axes. --
                  -- Constraint: Precision unzi is specified in terms
                     of the following granule, if supplied; otherwise
                     in terms of the axis HMU defined for the schedule. --
      #IMPLIED    -- Default: when not specified on a wand or baton, #ALL.
                     When not specified on a modgrp, modscope, progrp, or
                     proscope inherits effective value of containing wand,
                     modgrp, baton or progrp. --

   portion        -- Portion modified or projected --
                  -- Modification or projection applies to entire
                     event, modscope or proscope (whole), or only to
                     the portions that are co-located with the
                     modifying modscope or projecting proscope
                     (part). --
      NAMES       -- Lextype: ("WHOLE"|"PART")+ --
                  -- Constraint: One specification for each axis or
                     one for all axes. --
      #IMPLIED    -- Default: when not specified on a wand or baton,
                     WHOLE. When not specified on a modgrp,
                     modscope,progrp, or proscope inherits effective
                     value of containing baton,modgrp, progrp, or
                     wand. --
>
<!entity % sched "INCLUDE">
]]><!-- rend -->
```

10.2 Object Modification

Although the semantics of objects, and therefore of their modification, is outside the scope of this International Standard, it can nevertheless be useful to specify the relationships between objects and their modifiers in a standardized way.

NOTE 316 In the same way, instructions for patching together a series of audio signal processing devices could be executed by a person who knows nothing about the effects they create.

NOTE 317 Although object modification is an optional facility of HyTime and requires support of the “modify” option, the modify option need not be supported by applications in which modification is merged into the application processing and is therefore not independently specifiable.

10.2.1 Object modifier

An object modifier affects the object of an event during its rendition.

NOTE 318 Obvious examples of object modifiers include color mapping functions and audio signal processing instructions.

Object modifiers are defined by applications, not by this standard.

NOTE 319 Useful sets of object modifiers can be defined by industry groups for public use.

An object modifier can be represented by any element type or data.

NOTE 320 In particular, it can be represented by information expressed in terms of a data content notation.

NOTE 321 An object modifier is not necessarily an element.

An object modifier is itself subject to object modification by object modifiers.

10.2.2 Direct association of modifiers (modifier rule)

The element form **modifier rule** (*modrule*) directly associates one or more events, event groups, event schedules, modscopes, modgrps, and wands with a modifier or modifier patch (see *10.2.4 Modifier Patch and Wand Patch*); the association indicates that the objects scheduled by the events (and scheduled by the events contained by the event groups and event schedules), or the modifiers scheduled by the modscopes (and scheduled by the modscopes contained by the modgrps and wands), are modified by the modifier or modifier patch. The attribute **events and event groups** (*events*) identifies the events, event groups, event schedules, modifier scopes, modifier groups, and wands whose objects and modifiers are to be modified. The attribute **modifier or modifier patch** (*modifier*) identifies the modifier or modifier patch to be applied.

```
                        <!-- Modifier Rule -->
<![ %modify; [
<!element
   modrule        -- Modifier rule --
                  -- Clause: 10.2.2 --
   - O
   (%HyCFC;)*

-- Attributes [rend]: modrule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!attlist
   modrule        -- Modifier rule --
                  -- Clause: 10.2.2 --

   events         -- Events and modscopes --
      CDATA       -- Reference --
                  -- Reftype:
                     (event|evgrp|evsched|modscope|modgrp|wand)* --
      #REQUIRED

   modifier       -- Modifier or modifier patch --
      CDATA       -- Reference --
                  -- Constraint: only one modifier or modifier
                     patch --
      #REQUIRED
>
<!entity % rend "INCLUDE">
]]><!-- modify -->
```

10.2.3 Association of modifiers by position in finite coordinate spaces

In HyTime, modifiers can occur in schedules called “wands”; these are associated with one or more event schedules by means of wand rules (see *10.2.3.1 Wand Rule*), optionally with the aid of wand patches (see *10.2.4 Modifier Patch and Wand Patch*). Within a wand, a modifier is scheduled by an event-like element called a “modifier scope”, in the same way that an information object is scheduled by an event. The extent of the modifier scope defines the region of the FCS wherein the objects scheduled by events contained in the associated event schedules (or modifiers scheduled by modscopes contained in associated wands) will be modified.

NOTE 322 A wand is a symbol of authority. Its well-known association with magic seems appropriate, considering that modification semantics are undefined by HyTime. This magical connotation is in contrast with the baton construct of the projection facility (see *10.3 Projection*), which, though also a symbol of authority, has semantics defined by HyTime.

10.2.3.1 Wand Rule

The element form **wand rule** (*wandrule*) associates one or more event schedules (whose objects are to be modified) and/or wands (whose modifiers are to be modified) with a modifying wand or wand patch (see *10.2.4 Modifier Patch and Wand Patch*). The modifiers scheduled by the modifying wand or wand patch are applied to the objects and modifiers scheduled by the event schedules and wands, insofar as the extents of the modscopes contained in the modifying wand are co-located with the extents of the events and modscopes of those event schedules and wands.

NOTE 323 The co-location criteria required to trigger modification can be flexibly specified using the select and portion attributes of the modifying wands, modgrps, and modscopes (see *10.1.1 Precision of Selection*).

The attribute **event schedules** (*evscheds*) identifies the event schedules containing the events that schedule the objects to be modified and/or the wands containing the modscopes that schedule the modifiers to be modified; the attribute **wand or wand patch** (*wand*) identifies the modifying wand or wand patch.

NOTE 324 HyTime allows more than one wand or modifier rule to be used to specify the application of modification to the same object or modifier. Applications may choose to constrain such phenomena, and/or to define the effect of multiple modifications. However, a single wandrule can associate only one wand or wand patch. The use of wand patches makes the order in which the modifications are to be applied explicit.

NOTE 325 Although the wand rule form allows applications to associate different wands with different schedules and/or wands, there is no implication that all wands are interchangeable: they must be designed with the properties of the schedules and/or wands they are to modify in mind.

NOTE 326 Wand patches are discussed along with modifier patches in *10.2.4 Modifier Patch and Wand Patch*.

NOTE 327 The wand rules and modifier rules required for a rendition can be collected in a “rendition rule” (see *10.4 Rendition rule*).

In any rendition, all modifications specified for scheduled modifiers must be made before such scheduled modifiers are themselves applied.

NOTE 328 HyTime imposes no validation requirement that circular modification specifications, such as when a modifier modifies itself, be avoided in documents or hyperdocuments. It may be reasonable, however, for some kinds of HyTime engines to detect such loops, and for certain kinds of applications to support efforts intended to avoid, create, and/or manage such loops.

```
                          <!-- Wand Rule -->
<![ %wand; [
<!element
   wandrule       -- Wand rule --
                  -- Clause: 10.2.3.1 --
   - O
   (%HyCFC;)*

-- Attributes [rend]: wandrule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!attlist
   wandrule       -- Wand rule --
                  -- Clause: 10.2.3.1 --

   evscheds       -- Event schedules --
      CDATA       -- Reference --
                  -- Reftype: (evsched|wand)* --
                  -- Constraint: all evscheds and wands must be in
                     same logical FCS as all of the wands referenced
                     by the wand attribute. --
      #REQUIRED

   wand           -- Wand or wand patch --
      CDATA       -- Reference --
                  -- Reftype: (wand|wndpatch) --
      #IMPLIED    -- Default: none --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->
```

10.2.3.2 Wand

The element form **wand** (*wand*) is a schedule for modifier scopes.

The modifier scopes in a wand cannot overlap on any coordinate axis. However, it is permissible that there be gaps along the axes in which no modification is specified.

NOTE 329 This rule prevents situations in which the order in which overlapping modifications should be applied would be ambiguous.

```
                            <!-- Wand -->
<![ %wand; [
<!element
   wand           -- Wand --
                  -- Clause: 10.2.3.2 --
   - O
   (modgrp|modscope)+

-- Attributes [sched]: sched --
-- Attributes [rend]: select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, dimref:schdspec,
   pdimref:prjtarg, pulsemap:pulsemap, rfcsloc:prjsrc --
-- Referrers [rend]: batrule:scheds, batrule:targschd,
   prorule:sources, prorule:targschd, wandrule:wand, wndpatch --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->
```

10.2.3.3 Modifier scope

The element form **modifier scope** (*modscope*) associates one or more extents with a modifier, in order to specify the modification of objects and modifiers that occur within those extents.

If a value is specified for the attribute **object modifier** (*modifier*), it identifies the object modifier(s) and/or modifier patches to be applied. If no value is specified, the modscope is either merely a “placeholder” used to affect the next specified modscope's extent by means of the implicit dimref facility (see *9.8.1 Implicit dimension reference*) or the modscope itself carries the modifier parameters and semantics directly, in some fashion defined by the application.

NOTE 330 If no value is specified for the value of the modifier attribute, the semantics of object modification could be conveyed directly by the element type, by the values of one or more additional attributes, by the content, or by any combination of the foregoing, as defined by the application. For example, the application might define a specific element type for use as a placeholder that carries no modification semantics.

```
                       <!-- Modifier Scope -->
<![ %wand; [
<!element
   modscope       -- Modifier scope --
                  -- Defines the scope of a modifier as one or more
                     extents. --
                  -- Clause: 10.2.3.3 --
   - O
   (%HyCFC;)*

-- Attributes [base]: overrun --
-- Attributes [sched]: exspec --
-- Attributes [rend]: modscope, select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!attlist
   modscope       -- Modifier scope --
                  -- Clause: 10.2.3.3 --

   modifier       -- Modifiers and modifier patches to be
                     applied --
      CDATA       -- Reference --
      #IMPLIED    -- Default: the modification semantics, if any, are
                     specified directly by the GI, attributes and/or
                     content of this element in some
                     application-defined fashion --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->
```

10.2.3.4 Modifier scope group

The element form **modifier scope group** (*modgrp*) is a container for a set of modifier scopes or other modifier scope group elements that are specified contiguously within a wand element. The elements it contains can be modscopes or other modgrps. These syntactic containers fulfill a role which corresponds precisely to that of event groups, and the same features, most notably the grpdex and grprepet attribute forms, are provided (see *9.4.3 Group extent specification*).

```
                    <!-- Modifier Scope Group -->
<![ %wand; [
<!element
   modgrp         -- Modifier scope group --
                  -- Clause: 10.2.3.4 --
   - O
   (modgrp|modscope)+

-- Attributes [rend]: select --
-- OptionalAttributes [sched]: grpdex, grprepet, pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->
```

10.2.4 Modifier Patch and Wand Patch

The term “patch” is applied collectively to the element forms **wand patch** (*wndpatch*) and **modifier patch** (*modpatch*). A modifier patch contains a list of references to modifiers and modifier patches; a wand patch contains a list of references to wands and wand patches. The order of references in the list determines, in general, the order in which the modifications that correspond to the referenced modifiers are applied to the objects.

NOTE 331 “In general” because the specification order of sets of modifiers that are applied in parallel is not significant.

NOTE 332 The term “patch” is borrowed from the traditional lexicon of analog electronic music. In that context, the term meant a particular sequence of connections between such devices as voltage-controlled amplifiers, oscillators, and filters, all usually accomplished with “patch cords”, to create a particular effect.

Serial and parallel processing are indicated in the list by a subset of the SGML model group syntax in which the “and” connector (&) represents parallel processing and the “seq” connector (,) represents serial.

NOTE 333 For example, in the case of audio effect processing, a patch of

```
(id1, id2, (id3 & (id4, id5)), id6)
```

means that the audio signal passes through id1 effects, then id2, then is directed to both id3 and id4, with the output of id4 going to id5. Finally, the output of both id3 and id5 is processed by id6.

The element form **wand patch** (*wndpatch*) is a patch in which all of the references are to wands or wand patches. Wndpatch elements are referenced in the values of the wand attributes of wandrule elements.

The element form **modifier patch** (*modpatch*) is a patch in which all of the references are to object modifiers or modifier patches. Modpatch elements are referenced in the values of the modpatch attributes of modrule elements.

NOTE 334 HyTime does not constrain patches that include loops or recursions, e.g., a patch that references itself. The validity of such a patch depends on the application.

```
                       <!-- Modifier Patch -->
<![ %patch; [
<!element
   modpatch       -- Modifier patch --
                  -- Clause: 10.2.4 --
   - O
   (#PCDATA)      -- Reference --
                  -- Note: IDREFs may refer to modpatch
                     elements. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: modpatch, modrule:modpatch --
>
<!entity % modify "INCLUDE">
]]><!-- patch -->

                         <!-- Wand Patch -->
<![ %wndpatch; [
<!element
   wndpatch       -- Wand patch --
                  -- Clause: 10.2.4 --
   - O
   (#PCDATA)      -- Reference --
```

```
                    -- Reftype: (wndpatch|wand)* --
                    -- Constraint: subset of SGML model group with
                       IDREFs as names and no occurrence indicators or
                       OR connectors. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: wandrule:wand, wndpatch --
>
<!entity % wand "INCLUDE">
]]><!-- wndpatch -->
```

10.3 Projection

Projection deals with the semantic copying of events, modifier scopes, and projector scopes, together with the objects, modifiers, and projectors they schedule, to event schedules, wands, and batons, respectively, perhaps transforming their positions and extents in the process. Events and modscopes whose objects and modifiers are subject to modification in their source contexts can be projected with or without that modification in effect.

NOTE 335 The semantic effects of projection on objects and modifiers are necessarily application defined, inasmuch as HyTime does not standardize the semantics of objects and modifiers.

NOTE 336 Although projection is an optional feature of HyTime and requires support of the “project” option, support for the “project” option is not needed for applications in which projection is merged into the application processing and is therefore not independently specifiable.

Projection can be from one schedule (evsched, wand, or baton) to another—the “source” schedule to the “target” schedule. The targets of the projected events, modscopes, and proscopes can be other schedules of the same FCS as the source schedules, or the schedules of other FCSs. If the source FCS and the target FCS(s) are not the same, the possibility exists for the measurement domains and/or scales of the axes to differ.

NOTE 337 For example, in the case of the rendition of a musical score, notes scheduled in terms of the virtual time can be projected onto schedules in another FCS measured in terms of real time. Another example is in the realm of project management, in which a sequence of tasks is arrayed along an axis of a visible graph intended to show a time line, which, when rendered (that is, when the project is executed), is projected onto a real time axis in another FCS. A third example is the projection of events that describe a geographic area in real space units, which is rendered as a map by proportional projection onto another FCS using much smaller real space units.

The source and target FCSs need not have the same number of coordinate axes. However, the source and target schedules must have the same number of axes, and the order of their axes are treated as corresponding to one another.

NOTE 338 In other words, the first axis of the source schedule corresponds to the first axis of the target schedule, the second to the second, and so on.

It is possible for events, modscopes and proscopes to be projected any number of times from one schedule to another. In the case of subsequent projections, the rendition in effect may become the source for another rendition.

NOTE 339 If there is no need for the semantic of renditional projection, it may be preferable to accomplish essentially the same end using the grprepet attributes of an evgrp, modgrp, or progrp to schedule the event, modscope or proscope multiple times.

NOTE 340 HyTime imposes no validation requirement that circular projection specifications be avoided in documents or hyperdocuments. It may be reasonable, however, for some kinds of HyTime engines to detect such loops, and for certain kinds of applications to support efforts intended to avoid, create, and/or manage such loops.

Target schedules can be the targets of projection from multiple source schedules, and they can also contain "directly" scheduled events, modscopes, and proscopes, that is, elements scheduled in the normal way, without the use of projection.

The objects scheduled by events (and the modifiers scheduled by modscopes) that were projected onto the target event schedules (or wands) are subject to any modifiers scheduled by modscopes contained in and/or projected onto wands that govern (by virtue of a wand rule specification) the objects (or modifiers) scheduled by such target event schedules (or wands). Similarly, projected events, modscopes, and proscopes are subject to further projection by projectors scheduled by proscopes contained in and/or projected onto batons that govern (by virtue of a baton rule specification) the events, modscopes, and proscopes scheduled by such target event schedules, wands and batons.

In HyTime, the specifications for projection are called "projectors"; they specify projection functions that determine the scheduled extents of the elements as projected onto the target schedules.

NOTE 341 Depending upon the nature of the document, the modes of specifying projections can vary widely. For example:

— A single projector could be fixed for the entire schedule; for example, when a sequence of graphics events is to be scaled uniformly. Alternatively, it could change from place to place; for example, in a slide show where the durations of some slides are sped up to create a special effect, or in music that slows down at the end of a piece.

— The scaling factor could be expressed precisely; for example, if an HMU corresponded to a frame in NTSC video, the time scaling factor (the "tempo") might be 1/30 second per HMU. Alternatively, the expression of the scaling factor could be imprecise (by using descriptive text, for example). In music, for example, it is common to specify tempos in such natural language terms as "slow", "moderate", and even "not too fast". A precise definition would eventually be supplied for an imprecise scaling factor at rendition time. (Any notation can be used to specify the projection functions, including notations that only human beings, such as musicians reading tempo markings in Italian, can be expected to interpret for best results.)

— A change to a scaling factor could occur over a period of space or time, in which case the rate of change and final scaling factor can be expressed as a projection function in the content of a projector using any appropriate notation. Such a projection function could be specified as a precise formula (for example, linear acceleration to 4 beats per second), or as imprecise instructions (for example, "speed up little by little to a moderate rate", or "zoom quickly to a tight close-up"). A precise definition would eventually be supplied for an imprecise projection function at rendition time.

By preserving source extents and their governing projectors, rather than just the extents of a given rendition of the document, the intention of the document creator is preserved.

NOTE 342 In other words, we know why different portions of the rendition lasted as long as they did (or occupied as much space as they did), and where it is acceptable to vary the extents in a subsequent rendition. For example, we can tell whether two events occurred simultaneously by chance, or whether the author intended that they be synchronized in all renditions.

This information can be valuable during subsequent editing of the information, or if portions of it are extracted for use elsewhere, as is common in hypermedia applications.

NOTE 343 Text processing and compound document page layout are outside the scope of this International Standard. Nonetheless, it can be useful for understanding projection to draw an analogy from text processing. In this analogy, unprojected extent is like an unformatted document, projected extent is like a formatted document, and projectors are like a style sheet that specifies how a formatted document can be produced from the unformatted. Depending upon the precision and completeness of the style sheet, only one formatted version of the document may be possible, or many different ones. However, projectors can be part of the document creator's intention as well as the expression of a stylistic interpretation. Therefore, they are representable in HyTime even though applications may also include them in style sheets whose values override those in the document, or define them more precisely.

10.3.1 Projector

The content of the element form **projector** (*projectr*) defines a projection function for the events, modscopes, and proscopes it projects.

The **projector notation** (*notation*) attribute specifies the notation used in the content. If no value is specified, the projector functions in the content are assumed to be expressed as a set of extent projector (expro) elements.

NOTE 344 The HyTime-defined “HyPro” projector notation may be used (see *10.3.1.1 HyTime Projector Notation*).

The attribute **strictness** (*strict*) is a character string that indicates the intended degree of adherence to the projection function. The values are application specific and affect only the application's execution of the calculated projection, not the calculation itself.

NOTE 345 For example, values might be “rubato” or “within 10 percent”.

NOTE 346 An application can build a library of projectors by creating descriptive text and description table definitions for commonly used values, e.g., “double” for 2:1 ratio (see *6.7.2 Descriptive text*).

```
                          <!-- Projector -->
<![ %project; [
<![ %HyPro; [
   <!entity % dpro "HyPro">
]]>
<!entity %
   dpro           -- Default projector notation --
                  -- Clause: 10.3.1 --

   "#IMPLIED"
>
<!element
   projectr       -- Projector --
                  -- Clause: 10.3.1 --
                  -- Defines a projection function for events,
                     modscopes, and proscopes in a source FCS. --
   O O
   (%HyCFC;|%dimlist;|dimpro|expro|extent|extlist)*
                  -- Constraint: If no extent projector function
                     notation is specified, content is restricted to
                     (expro)* --

-- Attributes [rend]: projectr --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: prorule:proseq, proscope:projectr, proseq --
>
<!attlist
   projectr       -- Projector --
                  -- Clause: 10.3.1 --

   notation       -- Projector notation --
      NAME        -- Lextype: NOTATION --
      %dpro;      -- Default: Content restricted to (expro)* --

   strict         -- Strictness --
                  -- Indication of the strictness with which the
                     projection function is intended to be
                     executed. --
      CDATA
      #IMPLIED    -- Default: No indication --
>
<!entity % rend "INCLUDE">
]]><!-- project -->
```

10.3.1.1 HyTime Projector Notation

The notation **HyTime Projector Notation** (*HyPro*) specifies projection functions implicitly by specifying the target extents on the target schedules into which the events, modscopes, and proscopes of the source schedules are projected. All of the extents of all of the source events, modscopes, and proscopes projected by the containing projectr are projected as a matrix, with each source extent being projected onto as many target extents as are specified. All projections of events, modscopes, and proscopes are done linearly.

The attribute **extent list notation** (*extlistn*) specifies the notation used to describe extent lists within HyPro notation. If no value is specified, the HyPro notation consists entirely of expro, dimpro, extent, and extlist elements. If a value is specified, the HyPro notation can also include extents specified in the notation named in the value.

NOTE 347 HyExtLst notation may be used as a HyPro extent list notation (see *9.4.6 HyTime extent list notation*).

NOTE 348 The HyPro notation is syntactically indistinguishable from HyExtLst when xpro and dimpro are not used and the value of the extlistn attribute is “HyExtLst”. Even in those circumstances, HyExtLst cannot be used to describe projectors because it is semantically different from HyPro; HyExtLst is used simply to specify extents, whereas HyPro is used to specify sets of projection functions derived from the extents it specifies (and from the source extents to be projected).

```
                  <!-- HyTime Projector Notation -->
<![ %HyPro; [
<![ %HyExtLst; [
   <!entity % dpextlst "HyExtLst">
]]>
<!entity %
   dpextlst       -- Default projector extent list notation --
                  -- Clause: 10.3.1.1 --

   "#IMPLIED"
>
<!notation
   HyPro          -- HyTime Projector Notation --
                  -- Clause: 10.3.1.1 --
                  -- Defines projection functions by deriving linear
                     relationships between each dimension of each of
                     the specified extents (in the target FCS) and its
                     corresponding dimension in the source FCS.
                     Projector functions for individual extents or
                     dimensions may also be specified using expro or
                     dimpro elements. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Projector Notation//EN"

-- Attributes [rend]: HyPro --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyPro          -- HyTime Projector Notation --
                  -- Clause: 10.3.1.1 --
   extlistn       -- Extent list notation --
                  -- Notation used to specify the target extents. --
      NAME        -- Lextype: NOTATION --
      %dpextlst;  -- Default: Only expro, dimpro, extent, and extlist
                     elements allowed. --
>
<!entity % project "INCLUDE">
]]><!-- HyPro -->
```

10.3.1.2 Extent Projector

The content of the element form **extent projector** (*expro*) defines projection functions for projection of the events, modscopes, and proscopes it projects to a single target extent.

The **extent projector notation** (*notation*) attribute specifies the notation used in the content. If no value is specified, the projector functions in the content are assumed to be expressed as a set of dimension projector (dimpro) elements.

NOTE 349 The HyTime-defined “HyExPro” extent projector notation may be used (see *10.3.1.3 HyTime Extent Projector Notation*).

```
                      <!-- Extent Projector -->
<![ %project; [
<![ %HyExPro; [
   <!entity % dexpro "HyExPro">
]]>
<!entity %
   dexpro         -- Default extent projector notation --
                  -- Clause: 10.3.1.2 --

   "#IMPLIED"
>
<!element
   expro          -- Extent Projector --
                  -- Clause: 10.3.1.2 --
                  -- Defines projection functions to a single target
                     extent for extents in a source FCS. --
   O O
   (%HyCFC;|%dimlist;|dimpro)*
                  -- Constraint: If no extent projector notation is
                     specified, content is restricted to (dimpro)* --

-- Attributes [rend]: expro --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   expro          -- Extent Projector --
                  -- Clause: 10.3.1.2 --

   notation       -- Extent projector notation --
      NAME        -- Lextype: NOTATION --
      %dexpro;    -- Default: Content restricted to (dimpro)* --
>
<!entity % rend "INCLUDE">
]]><!-- project -->
```

10.3.1.3 HyTime Extent Projector Notation

The notation **HyTime Extent Projector Notation** (*HyExPro*) specifies projection functions implicitly by specifying the dimensions of a single target extent on the target schedule. All of the extents of all of the source events, modscopes, and proscopes projected by the containing expro element are projected linearly onto this single extent.

The attribute **extent specification notation** (*exspnot*) specifies the notation used to describe the target extent. If no value is specified, the HyExPro notation consists entirely of dimpro and dimspec elements.

NOTE 350 HyExSpec notation can be used (see *9.4.4 Scheduled extent*).

```
              <!-- HyTime Extent Projector Notation -->
<![ %HyExPro; [
<![ %HyExSpec; [
   <!entity % dpexspec "HyExSpec">
]]>
<!entity %
   dpexspec       -- Default projector extent specification
                     notation --
                  -- Clause: 10.3.1.3 --

   "#IMPLIED"
>
<!notation
   HyExPro        -- HyTime Extent Projector Notation --
                  -- Clause: 10.3.1.3 --
                  -- Defines projection functions by deriving linear
                     relationships between each dimension of the
                     specified extent (in the target FCS) and its
                     corresponding dimension in the source FCS.
                     Projector functions for particular dimensions may
                     also be specified using dimension projector
                     elements. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Extent Projector Notation//EN"

-- Attributes [rend]: HyExPro --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyExPro        -- HyTime Extent Projector Notation --
                  -- Clause: 10.3.1.3 --

   exspnot        -- Extent specification notation --
                  -- Notation used to specify the target extent. --
      NAME        -- Lextype: NOTATION --
      %dpexspec;  -- Default: Only dimpro and dimspec elements
                     allowed. --
>
<!entity % project "INCLUDE">
]]><!-- HyExPro -->
```

10.3.1.4 Dimension Projector

The content of the element form **dimension projector** (**dimpro**) defines projection functions for projection of the dimensions of the events, modscopes, and proscopes it projects to a single dimension on a single axis of the target schedule.

The **dimension projector notation** (*notation*) attribute specifies the notation used in the content. A value is required for this attribute.

NOTE 351 Signed nonzero integers, dimrefs, and markfuns (axis markers and elements that resolve to axis markers) are permitted in the content by virtue of the %marklist parameter entity in the content model, but the markers themselves do not constitute an implicit notation. How to interpret the markers would not be known without specifying a notation using the *notation* attribute; this is the reason why the notation must be known to the HyTime engine.

NOTE 352 HyDimPro notation can be used (see *10.3.1.5 HyTime Dimension Projector Notation*).

```
                     <!-- Dimension Projector -->
<![ %project; [
<![ %HyDimPro; [
   <!entity % ddimpro "HyDimPro">
]]>
<!entity %
   ddimpro        -- Default dimension projector notation --
                  -- Clause: 10.3.1.4 --

   "#REQUIRED"
>
<!element
   dimpro         -- Dimension projector --
                  -- Clause: 10.3.1.4 --
                  -- Defines a projection function for dimensions
                     along an axis of a source FCS. --
   O O
   (%HyCFC;|%marklist;)*

-- Attributes [rend]: dimpro --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dimpro         -- Dimension projector --
                  -- Clause: 10.3.1.4 --

   notation       -- Dimension projector notation --
      NAME        -- Lextype: NOTATION --
      %ddimpro;
>
<!entity % rend "INCLUDE">
]]><!-- project -->
```

10.3.1.5 HyTime Dimension Projector Notation

The notation **HyTime Dimension Projector Notation** (*HyDimPro*) specifies projection functions implicitly, by specifying a single target dimension on a single axis of the target schedule. All of the dimensions on the corresponding axis of all of the source events, modscopes, and proscopes projected by the containing dimpro element are projected linearly onto this single dimension of the target axis.

The attribute **dimension specification notation** (*dimspnot*) specifies the notation used to describe the target dimension. A value must be specified if HyDimSpc is not supported.

```
             <!-- HyTime Dimension Projector Notation -->
<![ %HyDimPro; [
<![ %HyDimSpc; [
   <!entity % dpdimspc "HyDimSpc">
]]>
<!entity %
   dpdimspc       -- Default projector dimension specification
                     notation --
                  -- Clause: 10.3.1.5 --

   "#REQUIRED"
>
<!notation
   HyDimPro       -- HyTime Dimension Projector Notation --
                  -- Clause: 10.3.1.5 --
                  -- Defines a projection function by deriving a
                     linear relationship between the specified
                     dimension (in the target FCS) and its
                     corresponding dimension in the source FCS. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Extent Projector Notation//EN"

-- Attributes [rend]: HyDimPro --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDimPro       -- HyTime Dimension Projector Notation --
                  -- Clause: 10.3.1.5 --

   dimspnot       -- Dimension specification notation --
                  -- Notation used to specify the target dimension. --
      NAME        -- Lextype: NOTATION --
      %dpdimspc;
>
<!entity % project "INCLUDE">
]]><!-- HyDimPro -->
```

10.3.2 Direct association of projectors

Events, event groups, event schedules, modifier scopes, modifier scope groups, wands, projector scopes, projector scope groups, and batons can all be directly associated with projectors by means of “projector rules.”

10.3.2.1 Projection of modified and unmodified objects

The attribute **project modified or unmodified objects** (*modified*) specifies whether the object scheduled by any source event (or the modifier scheduled by any source modscope) is modified by any modifiers to which it is subject in the source event schedule (or wand) when it is projected (“modify”), or whether the event's object (or the modscope's modifier) is projected without such modification (“unmodify”).

```
            <!-- Project modified or unmodified object -->
<![ %modify; %project; [
<!attlist
-- modified --    -- Project modified or unmodified object --
                  -- Clause: 10.3.2.1 --
   (batrule|prorule)

   modified       -- Project event (or modscope) with object (or
                     modifier) modified or unmodified --
                  -- Project modified objects (or modifiers)
                     ("modify"), or project objects (or modifiers)
                     without the modifications that are applied to
                     them ("unmodify") in their source schedules. --
                  -- Constraint: value is ignored for projected
                     elements that are not events or modscopes. --
      (modify|unmodify)
      modify
>
]]><!-- modify, project -->
```

10.3.2.2 Projector Rule

The element form **projector rule** (*prorule*) directly associates one or more events, event groups, event schedules, modscopes, modgrps, wands, proscopes, progrps, and batons with a projector (see *10.3.1 Projector*) or projector sequence (see *10.3.2.3 Projector sequence*). The association indicates that the events, modscopes, and proscopes consisting of and/or contained by the elements specified by the **projection sources** (*sources*) attribute are projected by the projectors specified by the **projectors and projector sequences** (*projectr*) attribute onto the schedules specified by the **target schedules** (*targschd*) attribute. Events will only be projected onto event schedules, modscopes will only be projected onto wands, and proscopes will only be projected onto batons.

NOTE 353 A combination of events, modscopes, and proscopes can be projected with a single prorule onto their respective appropriate schedules.

```
                       <!-- Projector Rule -->
<![ %project; [
<!element
   prorule        -- Projector rule --
                  -- Clause: 10.3.2.2 --
                  -- Direct association between sources, projectors,
                     and target schedules. --
   - O
   (%HyCFC;)*

-- Attributes [rend]: modified, prorule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: pdimref:prjby, rfcsloc:prjby --
-- Referrers [rend]: rendrule --
>
<!attlist
   prorule        -- Projector rule --
                  -- Clause: 10.3.2.2 --

   sources        -- Projection sources --
                  -- Sources for projection --
      CDATA       -- Reference --
                  -- Reftype: (baton|event|evgrp|evsched|modgrp|
                             modscope|progrp|proscope|wand)* --
      #REQUIRED

   projectr       -- Projectors and projector sequences --
                  -- Projectors and/or projector sequences to be used
                     for projection from sources to targets. --
      CDATA       -- Reference --
                  -- Reftype: (projectr|proseq)* --
      #REQUIRED

   targschd       -- Target schedules --
                  -- Schedules onto which sources will be projected by
                     projectors and/or projector sequences. --
      CDATA       -- Reference --
                  -- Reftype: (evsched|wand|baton)+ --
                  -- Constraint: Only events will be projected onto
                     event schedule targets; only modscopes will be
                     projected onto wands, and only proscopes will be
                     projected onto batons. --
      #REQUIRED
>
<!entity % rend "INCLUDE">
]]><!-- project -->
```

10.3.2.3 Projector sequence

The element form **projector sequence** (*proseq*) contains a list of IDs of projectors or projector sequences. The order of unique identifiers in the list determines the order in which the identified elements are applied.

```
                     <!-- Projector sequence -->
<![ %proseq; [
<!element
   proseq         -- Projector sequence --
                  -- Clause: 10.3.2.3 --
   - O
   (#PCDATA)      -- Reference --
                  -- Reftype: (proseq|projectr)* --
                  -- Constraint: subset of SGML model group with
                     IDREFs as names and no occurrence indicators, OR
                     connectors, or AND connectors. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: prorule:proseq, proseq --
>
<!entity % project "INCLUDE">
]]><!-- proseq -->
```

10.3.3 Association of projectors by position in finite coordinate spaces

Projectors can be scheduled in FCSs by means of “batons”. Batons are like event schedules; they contain projector scopes and projector scope groups, which are like events and event groups, respectively. Projector scopes contain projectors, in much the same way that events contain information objects. The extent of the projector scope is the nominal region of the FCS which contains the events, modscopes, and/or proscopes that are projected by the projector it contains. The events, modscopes, and proscopes within the region that are projected are limited to those scheduled by (or projected onto) the schedules specified by one or more “baton rules” that associate the source schedules, target schedules, and the batons that schedule the projectors that specify the projection functions.

10.3.3.1 Baton rule

The element form **baton rule** (*batrule*) associates one or more schedules (event schedules, wands, and/or batons) with a baton or baton sequence that governs their projection, and with the resulting target schedules. The attribute **schedules** (*scheds*) identifies the schedules; the attribute **baton** (*baton*) identifies the baton or baton sequence.

The attribute **target schedules** (*targschd*) identifies one or more schedules in the coordinate spaces onto which events affected by the baton are to be projected. The events, modscopes, and proscopes in all of the source schedules will be projected into all of the target schedules.

NOTE 354 A baton does not create schedules; it populates the target schedules associated with it by the baton rule. During rendition, the HyTime engine maintains an internal form of each target schedule that contains the projected events, modscopes or proscopes, together with any events, modscopes, or proscopes that may have been directly scheduled in the schedule.

NOTE 355 Although the baton rule form allows applications to associate different batons with different schedules there is no implication that all batons are interchangeable: they must be designed with the properties of the schedules they are to affect in mind.

NOTE 356 The baton rules required for a rendition can be collected in a “rendition rule” (see *10.4 Rendition rule*).

In any rendition, all projections specifying a given target schedule must be completed before projections that use that target schedule as a projection source are made.

```
                         <!-- Baton Rule -->
<![ %baton; [
<!element
   batrule        -- Baton rule --
                  -- Clause: 10.3.3.1 --
   - O
   (%HyCFC;)*

-- Attributes [rend]: batrule, modified --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!attlist
   batrule        -- Baton rule --
```

```
                  -- Clause: 10.3.3.1 --

   scheds         -- Schedules --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)* --
      #REQUIRED

   baton          -- Baton --
      CDATA       -- Reference --
                  -- Reftype: (baton|batonseq) --
      #REQUIRED

   targschd       -- Target schedules --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)+ --
      #REQUIRED
>
<!entity % project "INCLUDE">
]]><!-- baton -->
```

10.3.3.2 Baton

The element form **baton** (*baton*) is a schedule of projector scopes that, when a baton rule so specifies, governs the projection of events, modscopes, and proscopes semantically present in one or more schedules. A baton must be a schedule of the same semantic FCS as the event schedules, wands, and batons it affects.

NOTE 357 The baton element, like the baton of an orchestra conductor, controls the rendition of the document (hence the name). The name also reflects the fact that the projection model of HyTime was originally derived from the handling of duration and tempo in music.

The projector scope occurrences in a baton may overlap on any or all coordinate axes, and there can also be gaps between them in which no events, modscopes, or proscopes will be projected.

```
                            <!-- Baton -->
<![ %baton; [
<!element
   baton          -- Baton --
                  -- Clause: 10.3.3.2 --
   - O
   (progrp|proscope)+

-- Attributes [sched]: sched --
-- Attributes [rend]: select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, dimref:schdspec, pdimref:prjby,
   pdimref:prjtarg, pulsemap:pulsemap, rfcsloc:prjby,
   rfcsloc:prjsrc --
-- Referrers [rend]: batonseq, batrule:baton, batrule:scheds,
   batrule:targschd, prorule:sources, prorule:targschd --
>
<!entity % project "INCLUDE">
]]><!-- baton -->
```

10.3.3.3 Projector scope

The element form **projector scope** (*proscope*) associates one or more extents with a projector, in order to specify the projection of events, modscopes and proscopes that occur within those extents.

The attribute **scheduled projectors** (*projectr*) refers to the projectors scheduled by the projector scope.

NOTE 358 The semantic of having multiple extents for a single proscope is defined by the projector or projectors scheduled by the proscope.

```
                       <!-- Projector Scope -->
<![ %baton; [
<!element
   proscope       -- Projector scope --
                  -- Clause: 10.3.3.3 --
                  -- Defines the scope of a projector as one or more
                     extents. --
   - O
   (%HyCFC;)*

-- Attributes [base]: overrun --
-- Attributes [sched]: exspec --
-- Attributes [rend]: proscope, select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, pdimref:prjby, rfcsloc:prjby,
   rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!attlist
   proscope       -- Projector scope --
                  -- Clause: 10.3.3.3 --

   projectr       -- Scheduled projectors --
      CDATA       -- Reference --
                  -- Reftype: projectr* --
      #IMPLIED    -- Default: none --
>
<!entity % project "INCLUDE">
]]><!-- baton -->
```

10.3.3.4 Projector scope group

The element form **projector scope group** (*progrp*) is a container for a set of projector scopes or other projector scope group elements that are specified contiguously in a schedule. These syntactic containers fulfill a role that corresponds precisely to that of event groups, and the same features, most notably the grpdex and grprepet attribute forms, are provided (see *9.4.3 Group extent specification*).

```
                    <!-- Projector Scope Group -->
<![ %baton; [
<!element
   progrp         -- Projector scope group --
                  -- Clause: 10.3.3.4 --
   - O
   (progrp|proscope)+

-- Attributes [rend]: select --
-- OptionalAttributes [sched]: grpdex, grprepet, pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, pdimref:prjby, rfcsloc:prjby,
   rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!entity % project "INCLUDE">
]]><!-- baton -->
```

10.3.3.5 Baton sequence

The element form **baton sequence** (*batonseq*) contains a list of IDs of batons or baton sequences. The order of unique identifiers in the list determines the order in which the identified elements are applied. Using a baton sequence is exactly like using a sequence of projections from one FCS to a second, from the second FCS to a third, etc., but without the intervening FCSs.

```
                       <!-- Baton sequences -->
<![ %batonseq; [
<!element
   batonseq       -- Baton sequence --
                  -- Clause: 10.3.3.5 --
   - O
   (#PCDATA)      -- Reference --
                  -- Reftype: (batonseq|baton)* --
                  -- Constraint: subset of SGML model group with
                     IDREFs as names and no occurrence indicators, OR
                     connectors, or AND connectors. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: batonseq, batrule:baton --
>
<!entity % baton "INCLUDE">
]]><!-- batonseq -->
```

10.4 Rendition rule

The element form **rendition rule** (*rendrule*) allows the modifier, wand, projector and baton rules of one or more renditions to be grouped together.

NOTE 359 For example, a rendition rule could represent the related renditions that comprise a given presentation of a document or hyperdocument. At processing time, the ID of the rendition rule would be specified to the application, either as a direct parameter, or indirectly by means of a style sheet or SGML link process definition.

```
                       <!-- Rendition Rule -->
<![ %rend; [
<!element
   rendrule       -- Rendition rule --
                  -- Clause: 10.4 --
   - O
   (%ArcCFC;)*    -- Reference --
                  -- Reftype:
                     (batrule|modrule|prorule|rendrule|wandrule)* --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!entity % sched "INCLUDE">
]]><!-- rend -->
```

11 Conformance

As for SGML, documents, applications, and systems may all conform to this International Standard. In addition, HyTime distinguishes validating HyTime engines from conforming HyTime applications and systems that do not provide complete validation of documents against the provisions of this International Standard.

11.1 Conforming HyTime document

If a HyTime document complies with all provisions of this International Standard and is a conforming SGML document as defined in ISO 8879, it is a conforming HyTime document.

NOTE 360 The provisions allow wide latitude in choosing which constructs of HyTime to support.

NOTE 361 Conformance to HyTime is independent of whether the document also conforms to any other document architecture.

11.1.1 Basic hyperlinking HyTime document

A conforming HyTime document is a basic hyperlinking HyTime document if its HyTime declarations are as follows:

```
<!ATTLIST #NOTATION HyTime
   ArcFormA NAME     HyTime
   ArcNamrA NAME     HyNames
   ArcSuprA NAME     sHyTime
   ArcIgnDA NAME     HyIgnD
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    "HyTime"
   ArcQuant CDATA    #FIXED "NAMELEN 9"
   ArcDataF NAME     #FIXED HyBridN
   ArcBridF NAME     #FIXED HyBrid
   ArcAuto  (ArcAuto|nArcAuto) nArcAuto
   ArcOptSA NAMES    "GenArc base links locs rend sched"

-- Support attributes for HyTime only --
   hyqcnt   NUMBER   32
   GenArc   CDATA    "altreps included superdcn"
   base     CDATA    "bos"
   locs     CDATA    "nmsploc multloc refctl referatt refloc"
   links    CDATA    "hylink clink agglink traverse"
   exrefs   NAME     exrefs
   manyanch NUMBER   2
>
```

11.1.2 Basic scheduling HyTime document

A conforming HyTime document is a basic scheduling HyTime document if its HyTime declarations are as follows:

```
<!ATTLIST #NOTATION HyTime
   ArcFormA NAME     HyTime
   ArcNamrA NAME     HyNames
   ArcSuprA NAME     sHyTime
   ArcIgnDA NAME     HyIgnD
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    "HyTime"
   ArcQuant CDATA    #FIXED "NAMELEN 9"
   ArcDataF NAME     #FIXED HyBridN
   ArcBridF NAME     #FIXED HyBrid
   ArcAuto  (ArcAuto|nArcAuto) nArcAuto
   ArcOptSA NAMES    "GenArc base links locs rend sched"

-- Support attributes for HyTime only --
   hyqcnt   NUMBER   32
   GenArc   CDATA    "altreps included superdcn"
   base     CDATA    "bos desctxt dimspec HyFunk markfun"
   locs     CDATA    "nmsploc multloc refctl"
   links    CDATA    "agglink clink hylink traverse"
   sched    CDATA    "dimref fcsloc HyExSpec HyExtLst measure"
   exrefs   NAME     exrefs

   -- Constraint: may support more than 3 axes, but must support
      at least 3 to be a basic scheduling document. --
   manyaxes NUMBER   3
>
```

11.1.3 Minimal HyTime document

A conforming HyTime document is a minimal HyTime (and hyperlinking) document if its HyTime declarations are as follows.

NOTE 362 A minimal HyTime document is also a minimal hyperlinking document because a truly minimal HyTime document would provide no facilities over and above the General Architecture because HyTime has no required facilities. The inclusion of the clink facility essentially provides HyTime-defined semantics for the form of hyperlink already used in a majority of SGML documents, ID-based cross reference within a single document.

```
<!ATTLIST #NOTATION HyTime
   ArcFormA NAME     HyTime
   ArcNamrA NAME     HyNames
   ArcSuprA NAME     sHyTime
   ArcIgnDA NAME     HyIgnD
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    "HyTime"
   ArcQuant CDATA    #FIXED "NAMELEN 9"
   ArcDataF NAME     #FIXED HyBridN
   ArcBridF NAME     #FIXED HyBrid
   ArcAuto  (ArcAuto|nArcAuto) nArcAuto
   ArcOptSA NAMES    "GenArc base links locs rend sched"

-- Support attributes for HyTime only --
   hyqcnt   NUMBER   32
   links    CDATA    "clink"
   exrefs   NAME     nexrefs
>
```

11.1.4 Minimal scheduling HyTime document

A conforming HyTime document is a minimal scheduling HyTime document if its HyTime declarations are as follows:

```
<!ATTLIST #NOTATION HyTime
   ArcFormA NAME     HyTime
   ArcNamrA NAME     HyNames
   ArcSuprA NAME     sHyTime
   ArcIgnDA NAME     HyIgnD
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    "HyTime"
   ArcQuant CDATA    #FIXED "NAMELEN 9"
   ArcDataF NAME     #FIXED HyBridN
   ArcBridF NAME     #FIXED HyBrid
   ArcAuto  (ArcAuto|nArcAuto) nArcAuto
   ArcOptSA NAMES    "GenArc base links locs rend sched"

-- Support attributes for HyTime only --
   hyqcnt   NUMBER   32
   sched    CDATA    "sched dimref"
   manyaxes NUMBER   1
   exrefs   NAME     nexrefs
>
```

11.2 Conforming HyTime application

If a HyTime application meets the requirements of this sub-clause and is a conforming SGML application as defined in ISO 8879, it is a conforming HyTime application.

NOTE 363 A HyTime application could be specified by a document application profile of any other document architecture.

11.2.1 Application conventions

A conforming HyTime application's conventions can affect only areas that are left open to specification by applications.

NOTE 364 Some examples are: names of element types conforming to HyTime architectural forms, and substitute attribute names.

11.2.2 Conformance of documents

A conforming HyTime application shall require its documents to be conforming HyTime documents, and shall not prohibit any markup that this International Standard would allow in such documents.

NOTE 365 For example, an application markup convention could recommend that only certain minimization functions be used, but could not prohibit the use of other functions if they are allowed by the formal specification.

11.2.3 Conformance of documentation

A conforming HyTime application's documentation shall meet the requirements of this International Standard (see *11.5 Documentation requirements*).

11.3 Conforming HyTime system

If a HyTime system meets the requirements of this sub-clause and is a conforming SGML system as defined in ISO 8879, it is a conforming HyTime system.

NOTE 366 Whether a system is a conforming HyTime system is not affected by whether it is also a conforming implementation of a document architecture.

11.3.1 Conformance of documentation

A conforming HyTime system's documentation shall meet the requirements of this International Standard (see *11.5 Documentation requirements*).

11.3.2 Conformance to HyTime system declaration

A conforming HyTime system shall be capable of processing any conforming HyTime document that is not inconsistent with the system's HyTime system declaration (see *11.6 HyTime system declaration*). A conforming HyTime system shall report every unsupported HyTime facility used in any HyTime document it processes.

NOTE 367 Depending on the application, the designer should consider issuing a warning to the effect that because the required facility is not supported, the usefulness and reliability of the information may be affected, as well as the ability of the system to protect of the rights, safety, and other interests of information asset owners and users as expressed through the use of HyTime facilities.

NOTE 368 A system's inability to process data content notations that are not defined in this International Standard does not affect whether it is a conforming HyTime system.

11.3.3 Support for minimal HyTime documents

A conforming HyTime system shall be capable of processing a minimal HyTime document.

11.3.4 Application conventions

A conforming HyTime system shall not enforce application conventions as though they were requirements of this International Standard.

NOTE 369 Warnings of the violation of application conventions can be given, but they must be distinguished from reports of HyTime errors.

11.4 Validating HyTime engine

If a HyTime engine in a conforming HyTime system meets the requirements of this sub-clause, it is a validating HyTime engine.

NOTE 370 A conforming HyTime system need not have a validating HyTime engine. Implementors can therefore decide whether to incur the overhead of validation in a given system. A user whose multimedia authoring system allowed the validation and correction of HyTime documents, for example, would not require the validation process to be repeated when the documents are rendered by a processing system.

11.4.1 Error recognition

A validating HyTime engine shall find and report a reportable HyTime error if one exists, and shall not report an error when none exists.

A validating HyTime engine can optionally report other errors.

NOTE 371 This International Standard does not specify how a HyTime error should be handled, beyond the requirement for reporting it. In particular, it does not state whether the erroneous information should be treated as data, and/or whether an attempt should be made to continue processing after an error is found.

NOTE 372 This International Standard does not prohibit a validating HyTime engine from reporting an error that this International Standard considers non-reportable if the engine is capable of doing so. Neither does it prohibit an engine from recovering from such errors, nor does it require an engine to report them when it is not performing validation.

A validating HyTime engine may warn of conditions that are potentially, but not necessarily, errors.

NOTE 373 For example, several events that occupy the same position in a coordinate space.

NOTE 374 HyTime architectural forms do not constrain the construction of document type definitions, only document instances. However, a validating HyTime engine can optionally report DTD constructs that would prevent the creation of a valid conforming instance, or that would allow the creation of a nonconforming instance.

11.4.2 Identification of HyTime messages

Reports of HyTime errors, including optional reports, shall be identified as HyTime messages in such a manner as to distinguish them clearly from all other messages, including warnings of potential HyTime errors.

11.4.3 Content of HyTime messages

A report of a HyTime error, including an optional report, shall state the nature and location of the error in sufficient detail to permit its correction.

NOTE 375 This requirement is worded to allow implementors maximum flexibility to meet their user and system requirements.

11.5 Documentation requirements

The objectives of this International Standard will be met most effectively if users, at all levels, are aware that HyTime documents conform to an International Standard that is independent of any application or engine. The documentation of a conforming HyTime system or application shall further such awareness.

NOTE 376 These requirements are intended to help users apply knowledge gained from one HyTime system to the use of other systems, not to inhibit a casual and friendly writing style.

11.5.1 Standard identification

Standard identification shall be in the natural language of the documentation.

Standard identification text shall be displayed prominently:

a) in a prominent location in the front matter of all publications (normally the title page and cover page);

b) on all identifying display screens of HyTime programs; and

c) in all promotional and training material.

For applications, the identification text is:

```
A HyTime application conforming to
International Standard ISO/IEC 10744 --
Hypermedia/Time-based Structuring Language
```

For systems, the identification text is:

```
A HyTime system conforming to
International Standard ISO/IEC 10744 --
Hypermedia/Time-based Structuring Language
```

The documentation for a conforming HyTime system shall include a HyTime system declaration (see *11.6 HyTime system declaration*).

11.5.2 Identification of HyTime constructs

The documentation shall distinguish HyTime constructs from application conventions and system functions, and shall identify the HyTime constructs as being part of the Hypermedia/Time-based Structuring Language.

NOTE 377 The objective of this requirement is for the user to be aware of which constructs are common to all HyTime systems, and which are unique to this one. This will reduce the experienced user's learning time for a new system or application.

This International Standard shall be cited as a reference for supported HyTime constructs that are not specifically documented for the system or application. For example, if, for simplicity's sake, only a subset of some function is presented (such as by omitting some of the options of the fcsloc element form), it shall be stated clearly that other options exist and can be found in this International Standard.

11.5.3 Terminology

All HyTime constructs shall be introduced using the terminology of this International Standard, translated to the national language used by the publication or program.

Such standard terminology should be used throughout the documentation. If, notwithstanding, a non-standard equivalent is used for a standard term, it must be introduced in context and it shall not conflict with any standard HyTime terms, including terms for unsupported or undocumented constructs.

11.6 HyTime system declaration

A HyTime system declaration specifies the version of HyTime and all of the modules and optional facilities that a HyTime system can support. It is represented by a set of HyTime support declarations in the reference concrete syntax. There must be one option attribute for each supported module, within which there can be no duplication of option names. The module and option names appear in the system declaration in the same order that they appear listed in this International Standard.

The system declaration for a system that supports all of the modules and optional facilities of HyTime is:

```
<!NOTATION
   HyTime         -- HyTime Architecture --
                  -- A base architecture used in conformance with the
                     Architectural Form Definition Requirements of
                     International Standard ISO/IEC 10744. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!ATTLIST #NOTATION HyTime
   GenArc         -- General architecture facilities --
      CDATA       -- Lextype: csname+ --
      "altreps dafe dvlist HyLex HyOrd included ireftype lextype
       opacity REGEX superdcn"

   base           -- Base module facilities --
      CDATA       -- Lextype: csname+ --
      "activity actypes bos bosspec conloc desctxt dimspec HyDimLst
       HyDimSpc HyFunk markfun valueref"

   locs           -- Location address module facilities --
                  -- Clause: 6.3 --
      CDATA       -- Lextype: csname+ --
      "agrovdef anchloc bibloc dataloc datatok grovplan linkloc
       listloc mixedloc multloc nameloc nmsploc pathloc pgrovdef
       proploc queryloc refctl referatt refloc reftype relloc spanloc
       treecom treeloc treetype"

   links          -- Hyperlinks module facilities --
                  -- Clause: 6.3 --
      CDATA       -- Lextype: csname+ --
      "agglink clink hylink ilink traverse varlink"

   sched          -- Scheduling module facilities --
                  -- Clause: 6.3 --
      CDATA       -- Lextype: csname+ --
      "calibrat calspec dimref fcsloc grpdex grprepet HyCalSpc
       HyExSpec HyExtLst HyGrand measure objalign pulsemap sched"

   rend           -- Rendition module facilities --
                  -- Clause: 6.3 --
      CDATA       -- Lextype: csname+ --
      "baton batonseq HyDimPro HyExPro HyPro modify obextent patch
       project proseq wand wndpatch"

   anysgml        -- Any SGML declaration allowed --
                  -- Clause: 6.3 --
      (anysgml|nanysgml)
      nanysgml

   exrefs         -- External references allowed --
                  -- Clause: 6.3 --
      (exrefs|nexrefs)
      exrefs
   refmodel       -- SGML model groups for reftype --
                  -- Clause: 6.3 --
      (SGMLmdl|nSGMLmdl)
      nSGMLmdl

   hyqcnt         -- Highest quantum count limit --
                  -- Clause: 6.3 --
      NUMBER      -- Constraint: power of 2 >= 32  --
      32

   manyanch       -- Maximum number of anchors allowed in
                     hyperlinks --
                  -- Clause: 6.3 --
      NUMBER      -- Constraint: must be >= 2 --
      #IMPLIED    -- Default: no limit --

   manyaxes       -- Maximum number of axes allowed in coordinate
                     spaces --
                  -- Clause: 6.3 --
      NUMBER      -- Constraint: must be >= 1 --
      #IMPLIED    -- Default: no limit --
>
```

Annex A
(normative)

SGML Extended Facilities

A.1 Introduction

This normative annex defines the SGML Extended Facilities, a group of extensions to the functionality of SGML that are required for many hypertext and hypermedia applications. These facilities can be used independently of one another and with or without the HyTime facilities described in the body of this International Standard. They are:

— *A.2 Lexical Type Definition Requirements (LTDR)*

This clause contains the requirements for creating formal definitions of the lexical types used in property sets, lexical type constraints on element attributes and content, queries, and data tokenizer grove construction processes.

— *A.3 Architectural Form Definition Requirements (AFDR)*

This clause states the requirements for the formal definition of the architectural forms by which an enabling document architecture governs the SGML representation of its documents.

— *A.4 Property Set Definition Requirements (PSDR)*

This clause contains the requirements for creating formal definitions of the results of a parsing process, known as “property sets”. It also describes the basic concepts of “groves”, sets of tree structures constructed from the parsing results.

— *A.5 General Architecture*

This clause defines the architectural forms of the General Architecture, a set of extensions to SGML parsing functionality.

— *A.6 Formal System Identifier Definition Requirements (FSIDR)*

This clause states the requirements for the formal definition of notations used in system identifiers.

— *A.7 SGML Property Set*

This clause contains the SGML Property Set Definition Document, including the optional modules for the SGML Extended Facilities.

Applications that support any of these facilities are referred to generically as SGML Extended Facilities applications.

A.1.1 Conformance

Some facilities define their own conformance requirements and methods of indicating their use and their options, if any. In addition, the use of any of the facilities must conform to the following general requirements.

A.1.1.1 Application conventions

A conforming SGML Extended Facilities application's conventions can affect only areas that are left open by these facilities to specification by applications.

A.1.1.2 Conformance of documents

A conforming SGML Extended Facilities application shall require its documents to be conforming SGML documents, and shall not prohibit any markup that this International Standard would allow in such documents.

NOTE 378 For example, an application markup convention could recommend that only certain minimization functions be used, but could not prohibit the use of other functions if they are allowed by the formal specification.

A.1.1.3 Conformance of documentation

A conforming SGML Extended Facilities system's documentation shall meet the requirements of this annex.

A.1.1.3.1 Standard identification

Standard identification shall be in the natural language of the documentation.

Standard identification text shall be displayed prominently:

a) in a prominent location in the front matter of all publications (normally the title page and cover page);

b) on all identifying display screens of Extended Facilities programs; and

c) in all promotional and training material.

For applications, the identification text is:

```
An SGML Extended Facilities application conforming to
Annex A of International Standard ISO/IEC 10744 --
Hypermedia/Time-based Structuring Language
```

For systems, the identification text is:

```
An SGML Extended Facilities system conforming to
Annex A of International Standard ISO/IEC 10744 --
Hypermedia/Time-based Structuring Language
```

The documentation for a conforming SGML Extended Facilities system shall include the relevant system declarations or definition documents.

NOTE 379 For example, a storage manager definition document for Formal System Identifiers.

A.1.1.3.2 Identification of Extended Facilities constructs

The documentation shall distinguish Extended Facilities constructs from application conventions and system functions, and shall identify the Extended Facilities constructs as being part of the facility used.

NOTE 380 The objective of this requirement is for the user to be aware of which constructs are common to all Extended Facilities systems, and which are unique to this one. This will reduce the experienced user's learning time for a new system or application.

This International Standard and annex shall be cited as a reference for supported Extended Facilities constructs that are not specifically documented for the system or application. For example, if, for simplicity's sake, only a subset of some function is presented, it shall be stated clearly that other options exist and can be found in this International Standard.

A.1.1.3.3 Terminology

All Extended Facilities constructs shall be introduced using the terminology of this International Standard, translated to the national language used by the publication or program.

Such standard terminology should be used throughout the documentation. If, notwithstanding, a non-standard equivalent is used for a standard term, it must be introduced in context and it shall not conflict with any standard Extended Facilities terms, including terms for unsupported or undocumented constructs.

A.1.1.3.4 Application conventions

A conforming SGML Extended Facilities system shall not enforce application conventions as though they were requirements of this annex.

NOTE 381 Warnings of the violation of application conventions can be given, but they must be distinguished from reports of facility errors.

A.2 Lexical Type Definition Requirements (LTDR)

This clause states the requirements for the formal definition of lexical types. A “lexical type” is a pattern of characters to which a character string can conform. The pattern is called a “lexical model”. A lexical type declaration associates a name with a lexical model, defining a named lexical type.

Lexical types defined in accordance with these requirements are used in conjunction with the HyTime lexical model notation (HyLex; see *A.2.3 HyTime lexical model notation (HyLex)*) to prescribe the lexical characteristics of attribute values and of the data content of elements (see clause 5 and *A.5.4 Lexical types*). They are also used when searching a “match domain” for strings or token lists that satisfy a lexical pattern (see *A.4.4.2 Data tokenizer (DATATOK) grove construction*).

A.2.1 Lexical type set

Lexical types are declared within “lexical type sets”, along with “lexicographic orderings” (see *A.2.1.2 Lexicographic ordering*) and “lexical constraints” (see *A.2.1.3 Additional lexical constraints*). Lexical type declaration sets may also contain notation, data attribute definition list, and parameter entity declarations, each as defined in ISO 8879. Each declaration is preceded and followed by zero or more parameter separators, also as defined in ISO 8879.

Within a document, a lexical type use declaration identifies one or more lexical type declaration sets, whose lexical types are to be made available for use in the document. A lexical type use declaration for a lexical type set must precede uses of the lexical types declared within that set. There can be more than one lexical type use declaration in a DTD or LPD.

Syntactically, a lexical type use declaration is a processing instruction (PI), not an SGML markup declaration. Its syntax is (using the production syntax defined in ISO 8879):

[10] lexical type use declaration =
 pio,
 "IS10744",
 s+,
 "LEXUSE",
 (*ps+*,
 lexical type set entity name)+
 *ps**,
 pic

[11] lexical type set entity name =
name

where *name* is the name component of a *parameter entity name*.

Because lexical type set entities are never referenced within a DTD or LPD, they must be self-contained; notations, data attributes, parameter entities, lexicographic orderings, lexical constraints, and lexical types referenced within them must also be declared within them.

NOTE 382 A lexical type set may use parameter entity references to “import” declarations from other lexical type sets.

A.2.1.1 Lexical types

Lexical types are defined by “lexical type declarations”. A lexical type declaration takes the form of an SGML markup declaration with the following syntax:

[12] lexical type declaration =
mdo,
"LEXTYPE",
ps+,
lexical type name,
(*ps+*,
lexicographic ordering specification)?,
(*ps+*,
additional lexical constraint specification)*,
ps+,
lexical model specification,
*ps**,
mdc

[13] lexical type name =
name

[14] lexicographic ordering specification =
rni,
"ORDER",
lexicographic ordering name

where *lexicographic ordering name* is the name of a previously declared lexicographic ordering.

[15] additional lexical constraint specification =
rni,
"CHECK",
lexical constraint name

where *lexical constraint name* is the name of a previously declared additional lexical constraint.

[16] lexical model specification =
lexical model **|**
("SPEC",
ps+,
external identifier)

where

SPEC means that the *external identifier* identifies a lexical type definition, presumably written in a natural language.

[17] lexical model =
(*parameter literal* |
external identifier),
ps+,
notation name,
data attribute specification?

where the interpreted *parameter literal* or the data identified by the *external identifier* models a lexical type using the specified notation.

A.2.1.2 Lexicographic ordering

Characters in a computer system are inherently ordered by their coded representations, which assign them positions in a character set.

NOTE 383 The base-10 integer representation of a coded representation is called a “character number”. In an SGML document, characters can be represented by their coded representations (bit combinations) in the usual way, and also by means of a string containing a delimited character number, called a “numeric character reference”.

When characters are compared, as when sorting, testing conformance to a lexical type, or testing equality in a query, the character set order may not be suitable. A “lexicographic ordering” specifies an alternate set of character positions to be assigned for such comparisons.

NOTE 384 The alternative positions are sometimes called a “sort order”, although their use is not restricted to sorting. The lexicographic ordering definition is sometimes called a “sort translation”, although in fact no permanent character number translation occurs: the reordering is normally transitory and internal to the processing program.

Lexicographic orderings are identified by “lexicographic ordering declarations”. A lexicographic ordering declaration takes the form of an SGML markup declaration with the following syntax:

[18] lexicographic ordering declaration =
mdo,
"LEXORD",
ps+,
lexicographic ordering name,
ps+,
lexicographic ordering definition,
*ps**,
mdc

[19] lexicographic ordering name =
name

[20] lexicographic ordering definition =
lexicographic ordering expression |
("SPEC",
ps+,
external identifier)

where

SPEC means that the *external identifier* identifies a lexicographic ordering definition, presumably written in a natural language.

[21] lexicographic ordering expression =
(*parameter literal* |
external identifier),
ps+,
notation name,
data attribute specification?

where the interpreted *parameter literal* or the data identified by the *external identifier* expresses a lexicographic ordering using the specified notation.

A.2.1.3 Additional lexical constraints

Additional lexical constraints are identified by "lexical constraint declarations". A lexical constraint declaration takes the form of an SGML markup declaration with the following syntax:

[22] lexical constraint declaration =
mdo,
"LEXCON",
ps+,
lexical constraint name,
ps+,
lexical constraint definition,
*ps**,
mdc

[23] lexical constraint name =
name

[24] lexical constraint definition =
lexical constraint expression |
("SPEC",
ps+,
external identifier)

where

SPEC means that the *external identifier* identifies a lexical constraint definition, presumably written in a natural language.

[25] lexical constraint expression =
(*parameter literal* |
external identifier),
ps+,
notation name,
data attribute specification?

where the interpreted *parameter literal* or the data identified by the *external identifier* expresses a lexical constraint using the specified notation.

A.2.2 Lexical model notations

Any lexical model notation may be used to define lexical types. The following is a non-mandatory starter set of lexical modeling languages, one of which, HyLex, is then defined.

```
                <!-- HyTime Lexical Model Notation -->
<![ %HyLex; [
<!notation
   HyLex          -- HyTime lexical model notation --
                  -- Clause: A.2.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime lexical model notation (HyLex)//EN"

-- Attributes: HyLex --
-- CommonAttributes: altreps, included, superdcn --
>
<!attlist #NOTATION
   HyLex          -- HyTime lexical model notation --
                  -- Clause: A.2.3 --

   norm           -- Normalization --
      (norm|unorm)
      norm
>
]]><!-- HyLex -->

              <!-- POSIX Regular Expression Notation -->
<![ %REGEX; [
<!notation
   REGEX          -- POSIX regular expression notation --

   PUBLIC "ISO/IEC 9945-2:1997//NOTATION
           POSIX Regular Expression Notation//EN"
>
<!attlist #NOTATION
   REGEX          -- POSIX regular expression notation --

   case           -- Case sensitivity --
                  -- Case sensitive match (case) or case insensitive
                     match (icase) --
      (case|icase)
      case
>
]]><!-- REGEX -->
```

A.2.3 HyTime lexical model notation (HyLex)

This clause specifies the HyTime lexical model notation used in this International Standard and defines some useful instances of it.

A.2.3.1 Syntax

The syntax of a HyLex model is formally defined as that of an SGML content model, as specified in ISO 8879, conforming fully to the concrete syntax of lexical type sets, with the following differences:

— The HyLex model must be a model group, without inclusion or exclusion exceptions, but the grouping delimiters can be omitted if there is no occurrence indicator for the entire model group, or if the entire model group consists of a single lexical type name.

NOTE 385 Grouping delimiters are mandatory for subordinate model groups.

— An alternate form of subordinate model group, opened by DSO instead of GRPO, and closed by DSC instead of GRPC, defines a “match token model”. When HyLex is used to search data for a lexical pattern, those portions of the data that satisfy match token models are called “match tokens”, and are returned as the result of the search. When HyLex is used to prescribe the lexical type of a piece of data, a match token model has no effect other than that of a subordinate model group.

— Neither the HyLex model group nor its subordinate model groups can contain an AND connector.

— A content token can be a literal as defined in ISO 8879 or the name of a previously declared lexical type.

— Any content token or model group can be preceded by a reserved name indicator and the keyword “NOT”. A token or model group so qualified will match any character string except those that it would otherwise match.

NOTE 386 HyLex does not provide a way to limit the number of characters matched by a #NOT-qualified model.

— Any content token or model group not preceded by #NOT can be followed by an occurrence indicator, or it may be followed by a reserved name indicator and one or more of the following keywords:

ORDER This keyword and the following lexicographic ordering name specify a lexicographic ordering to be applied to data before matching it against the preceding content token or model group.

CHECK This keyword and the following additional lexical constraint name specify an additional lexical constraint to apply to the data matching the preceding content token or model group. Failing an additional lexical constraint has no effect on the matching process.

NOTE 387 While a content token or model group may not be directly followed by both an occurrence indicator and qualifiers, if a qualified token or model group is placed within a subordinate model group, the new subordinate model group can itself be followed by an occurrence indicator.

A.2.3.2 Normalized HyLex models

Normalized HyLex models employ a form of markup minimization intended for use when the lexical type to be defined is essentially a list of whitespace separated tokens. A “normalized” model can be converted to an equivalent unnormalized model by performing the following steps:

a) Insert “#ORDER SGMLCASE” as a qualifier after each literal content token.

b) Place each content token of the normalized model in its own match token model, including #ORDER and #CHECK qualifiers, but excluding occurrence indicators.

NOTE 388 For example, the normalized model “(NAME #CHECK ID,NUMBER+)” becomes “([NAME #CHECK ID],[NUMBER]+)”.

c) Insert an “s+” in between each pair of sequential subordinate model groups.

NOTE 389 For example, the normalized model “(NMTOKEN+,('#ANY',NUMBER)*)” becomes “([NMTOKEN]+,s+, (['#ANY' #ORDER SGMLCASE],s+,[NUMBER])*)”

d) Replace subordinate model groups followed by PLUS or REP occurrence indicators as follows:

— (submodel)+ becomes ((submodel),(s+,(submodel))*)

— [submodel]* becomes ([submodel]?,(s+,[submodel])*)

Note that if the original model was a match token model, the corresponding subordinate model groups in the replacement model are also match token models.

e) If the HyLex model (i.e. the top level model) is an OR group, turn it into a subordinate model group of a new sequential HyLex model.

f) Insert an “s*” at the beginning and end of the HyLex model.

NOTE 390 In the conventional comments that define lexical types for attributes and data content in this International Standard, “Lextype” signifies a HyLex model with the normalization attribute specified as “norm”, while “Ulextype” signifies an unnormalized HyLex model with the normalization attribute specified as “unorm” (see clause 5).

A.2.3.3 Intrinsic lexical types

HyLex is an SGML-aware lexical modeling language. As such, it defines a set of intrinsic SGML lexical types that are automatically available in any HyLex expression.

```
<!--
This file is identified by the following public identifier:

"ISO/IEC 10744:1997//NONSGML LTDR LEXTYPES SGML Lexical Types//EN"

Unless otherwise specified, all non-model lexical types and
lexicographic orderings are relative to the declared concrete syntax
of the document from which they are referenced.
-->

                <!-- HyTime Lexical Model Notation -->
<!NOTATION HyLex
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime lexical model notation (HyLex)//EN"
>
<!ATTLIST #NOTATION HyLex
   norm          -- Normalization --
      (norm|unorm)
      norm
>

                <!-- SGML lexicographic orderings -->

<!-- Note: For case-related ordering, the case rules that apply are
     the case rules of the document in which the lexicographic
     ordering (or lexical type that uses the lexicographic ordering)
     is used. -->

<!LEXORD
   SGMLCASE       -- SGML namecase substitution --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXORD Namecase substitution//EN"
>
<!LEXORD
```

```
   GENERAL        -- SGML general namecase substitution --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXORD
           General namecase substitution//EN"
>
<!LEXORD
   ENTITY         -- SGML entity namecase substitution --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXORD
           Entity namecase substitution//EN"
>
<!LEXORD
   RCSGENER       -- SGML reference concrete syntax general namecase
                     substitution --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXORD
           Reference concrete syntax general namecase
           substitution//EN"
>

                  <!-- SGML lexical constraints -->

<!LEXCON
   NAMELEN        -- SGML name length constraint --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON QUANTITY Name length//EN"
>
<!LEXCON
   PENTLEN        -- SGML parameter entity name length constraint --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON
           QUANTITY Parameter entity name length//EN"
>
<!LEXCON
   DTDORLPD       -- SGML DTD or LPD name --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON DTD or LPD name//EN"
>
<!LEXCON
   NOTATION       -- SGML Notation name --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON Notation name//EN"
>
<!LEXCON
   PARMENT        -- SGML Parameter entity name --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON Parameter entity name//EN"
>
```

```
<!LEXCON
   ENTITY         -- SGML General entity name --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON General entity name//EN"
>
<!LEXCON
   GI             -- SGML Generic identifier --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON Generic Identifier//EN"
>
<!LEXCON
   ID             -- SGML Unique identifier --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON Unique Identifier//EN"
>
<!LEXCON
   ATTNAME        -- SGML Attribute name --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXCON Attribute name//EN"
>
<!LEXCON
   compname       -- Property set component name --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXCON
           Property Set Component Name//EN"
>

                     <!-- SGML Lexical Types -->

<!LEXTYPE
   char           -- Character --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXTYPE Character//EN"
>

               <!-- SGML abstract character classes -->

<!LEXTYPE
   Digit          -- SGML digit --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE CLASS Digits (Digit)//EN"
>
<!LEXTYPE
   LCLetter       -- SGML lower-case letter --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Lower-case letters (LCLetter)//EN"
>
```

```
<!LEXTYPE
   Special        -- SGML special character --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Special minimum data characters (Special)//EN"
>
<!LEXTYPE
   UCLetter       -- SGML upper-case letter --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Upper-case letters (UCLetter)//EN"
>

               <!-- SGML concrete character classes -->

<!LEXTYPE
   NONSGML        -- Non-SGML characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Non-SGML characters (NONSGML)//EN"
>
<!LEXTYPE
   DATACHAR       -- SGML dedicated data characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Dedicated data characters (DATACHAR)//EN"
>
<!LEXTYPE
   DELMCHAR       -- SGML delimiter characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Delimiter characters (DELMCHAR)//EN"
>
<!LEXTYPE
   FUNCHAR        -- SGML inert function characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Inert function characters (FUNCHAR)//EN"
>
<!LEXTYPE
   LCNMCHAR       -- SGML lower-case name characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Lower-case name characters (LCNMCHAR)//EN"
>
<!LEXTYPE
   LCNMSTRT       -- SGML lower-case name start characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Lower-case name start characters (LCNMSTRT)//EN"
>
<!LEXTYPE
   MSICHAR        -- SGML markup-scan-in-characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Markup-scan-in-characters (MSICHAR)//EN"
>
<!LEXTYPE
   MSOCHAR        -- SGML markup-scan-out-characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Markup-scan-out-characters (MSOCHAR)//EN"
>
<!LEXTYPE
   MSSCHAR        -- SGML markup-scan-suppress characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Markup-scan-suppress characters (MSSCHAR)//EN"
>
<!LEXTYPE
   RE             -- SGML record end character --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Record end character (RE)//EN"
>
<!LEXTYPE
   RS             -- SGML record start character --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Record start character (RS)//EN"
>
<!LEXTYPE
   SEPCHAR        -- SGML separator characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Separator characters (SEPCHAR)//EN"
>
<!LEXTYPE
   SPACE          -- SGML space character --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Space character (SPACE)//EN"
>
<!LEXTYPE
   UCNMCHAR       -- SGML upper-case name characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Upper-case name characters (UCNMCHAR)//EN"
>
<!LEXTYPE
   UCNMSTRT       -- SGML upper-case name start characters --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           CLASS Upper-case name start characters (UCNMSTRT)//EN"
>

                       <!-- SGML delimiters -->

<!LEXTYPE
   AND            -- SGML and connector --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER And connector (AND)//EN"
>
<!LEXTYPE
   COM            -- SGML comment start or end --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Comment start or end (COM)//EN"
>
<!LEXTYPE
   CRO            -- SGML character reference open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Character reference open (CRO)//EN"
>
<!LEXTYPE
   DSC            -- SGML character reference open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Declaration subset close (DSC)//EN"
>
<!LEXTYPE
   DSO            -- SGML declaration subset open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Declaration subset open (DSO)//EN"
>
<!LEXTYPE
   DTGC           -- SGML data tag group close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Data tag group close (DTGC)//EN"
>
<!LEXTYPE
```

```
   DTGO           -- SGML data tag group open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Data tag group open (DTGO)//EN"
>
<!LEXTYPE
   ERO            -- SGML entity reference open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Entity reference open (ERO)//EN"
>
<!LEXTYPE
   ETAGO          -- SGML end-tag open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER End-tag open (ETAGO)//EN"
>
<!LEXTYPE
   GRPC           -- SGML group close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Group close (GRPC)//EN"
>
<!LEXTYPE
   GRPO           -- SGML group open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Group open (GRPO)//EN"
>
<!LEXTYPE
   LIT            -- SGML literal start or end --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Literal start or end (LIT)//EN"
>
<!LEXTYPE
   LITA           -- SGML literal start or end (alternative) --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Literal start or end (alternative) (LITA)//EN"
>
<!LEXTYPE
   MDC            -- SGML markup declaration close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Markup declaration close (MDC)//EN"
>
<!LEXTYPE
```

```
   MDO            -- SGML markup declaration open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Markup declaration open (MDO)//EN"
>
<!LEXTYPE
   MINUS          -- SGML exclusion --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Exclusion (MINUS)//EN"
>
<!LEXTYPE
   MSC            -- SGML marked section close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Marked section close (MSC)//EN"
>
<!LEXTYPE
   NET            -- SGML null end-tag --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Null end-tag (NET)//EN"
>
<!LEXTYPE
   OPT            -- SGML optional occurrence indicator --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Optional occurrence indicator (OPT)//EN"
>
<!LEXTYPE
   OR             -- SGML or connector --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Or connector (OR)//EN"
>
<!LEXTYPE
   PERO           -- SGML parameter entity reference open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Parameter entity reference open (PERO)//EN"
>
<!LEXTYPE
   PIC            -- SGML processing instruction close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Processing instruction close (PIC)//EN"
>
<!LEXTYPE
```

```
   PIO            -- SGML processing instruction open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Processing instruction open (PIO)//EN"
>
<!LEXTYPE
   PLUS           -- SGML required and repeatable; inclusion --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Required and repeatable; inclusion (PLUS)//EN"
>
<!LEXTYPE
   REFC           -- SGML reference close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Reference close (REFC)//EN"
>
<!LEXTYPE
   REP            -- SGML optional and repeatable --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Optional and repeatable (REP)//EN"
>
<!LEXTYPE
   RNI            -- SGML reserved name indicator --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Reserved name indicator (RNI)//EN"
>
<!LEXTYPE
   SEQ            -- SGML sequence connector --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Sequence connector (SEQ)//EN"
>
<!LEXTYPE
   SHORTREF       -- SGML short reference --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Short reference (SHORTREF)//EN"
>
<!LEXTYPE
   STAGO          -- SGML start-tag open --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Start-tag open (STAGO)//EN"
>
<!LEXTYPE
```

```
   TAGC           -- SGML tag close --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Tag close (TAGC)//EN"
>
<!LEXTYPE
   VI             -- SGML value indicator --

   SPEC
   PUBLIC "ISO 8879:1986//NOTATION LEXTYPE
           DELIMITER Value indicator (VI)//EN"
>

                 <!-- SGML modeled lexical types -->

<!LEXTYPE
   s              -- SGML "S" separator --

   "RE|RS|SEPCHAR|SPACE"
   HyLex [unorm]
>
<!LEXTYPE
   mindata        -- SGML minimum data --

   "(Digit|LCLetter|RE|RS|SPACE|Special|UCLetter)+"
   HyLex [unorm]
>
<!LEXTYPE
   nmchar         -- SGML name character --

   "Digit|LCNMCHAR|UCNMCHAR|nmstrt"
   HyLex [unorm]
>
<!LEXTYPE
   nmstrt         -- SGML name start character --

   "LCLetter|LCNMSTRT|UCLetter|UCNMSTRT"
   HyLex [unorm]
>
<!LEXTYPE
   csname         -- Case-sensitive name --

   "nmstrt,nmchar*"
   HyLex [unorm]
>
<!LEXTYPE
   NAME           -- SGML name --

   #ORDER GENERAL
   #CHECK NAMELEN

   "csname"
   HyLex [unorm]
>
<!LEXTYPE
```

```
   NAMES          -- SGML names --

   "NAME+"
   HyLex
>
<!LEXTYPE
   NUMBER         -- SGML number --

   #ORDER GENERAL
   #CHECK NAMELEN

   "Digit+"
   HyLex [unorm]
>
<!LEXTYPE
   NUMBERS        -- SGML numbers --

   "NUMBER+"
   HyLex
>
<!LEXTYPE
   NMTOKEN        -- SGML name token --

   #ORDER GENERAL
   #CHECK NAMELEN

   "nmchar+"
   HyLex [unorm]
>
<!LEXTYPE
   NMTOKENS       -- SGML name tokens --

   "NMTOKEN+"
   HyLex
>
<!LEXTYPE
   NUTOKEN        -- SGML number token --

   #ORDER GENERAL
   #CHECK NAMELEN

   "Digit,nmchar*"
   HyLex [unorm]
>
<!LEXTYPE
   NUTOKENS       -- SGML number tokens --

   "NUTOKEN+"
   HyLex
>

                <!-- SGML namespace lexical types -->

<!LEXTYPE
   ATTNAME        -- SGML attribute name --
   #CHECK ATTNAME

   "NAME"
   HyLex
>
<!LEXTYPE
   DTDORLPD       -- SGML document type or link type --

   #CHECK DTDORLPD

   "NAME"
   HyLex
>
<!LEXTYPE
   ENTITY         -- SGML entity name --

   #ORDER ENTITY
   #CHECK NAMELEN
   #CHECK ENTITY

   "nmstrt,nmchar*"
   HyLex [unorm]
>
<!LEXTYPE
   ENTITIES       -- SGML entity names --

   "ENTITY+"
   HyLex
>
<!LEXTYPE
   GI             -- SGML generic identifier --

   #CHECK GI

   "NAME"
   HyLex
>
<!LEXTYPE
   IDREF          -- SGML unique identifier reference --

   #CHECK ID

   "NAME"
   HyLex
>
<!LEXTYPE
   IDREFS         -- SGML unique identifier references --

   "IDREF+"
   HyLex
>
<!LEXTYPE
   NOTATION       -- SGML notation name --

   #CHECK NOTATION
   "NAME"
   HyLex
>
<!LEXTYPE
   PARMENT        -- SGML parameter entity name --

   #ORDER ENTITY
   #CHECK PENTLEN
   #CHECK PARMENT

   "nmstrt,nmchar*"
   HyLex [unorm]
>
<!LEXTYPE
   PENTITY        -- SGML parameter entity name prefixed by PERO --

   "PERO,PARMENT"
   HyLex [unorm]
>
<!LEXTYPE
   compname       -- Property set component name --

   #CHECK compname

   "NAME"
   HyLex [unorm]
>
<!LEXTYPE
   cnmlist        -- Property set component names --

   "compname+"
   HyLex
>

                  <!-- Other SGML lexical types -->

<!LEXTYPE
   fsi            -- Formal System Identifier --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXTYPE
           Formal System Identifier//EN"
>
<!LEXTYPE
   literal        -- SGML literal --

   "(LIT,[#NOT LIT],LIT)|(LITA,[#NOT LITA],LITA)"
   HyLex [unorm]
>
<!LEXTYPE
   attspecs       -- Attribute specifications --

   '(NAME,"=",(NMTOKEN|literal))*'
   HyLex
>
```

A.2.4 Lexicographic ordering definition notations

Any lexicographic ordering definition notation may be used to define lexicographic orderings. The following is a non-mandatory starter set consisting of one lexicographic ordering definition notation, HyOrd, which is then defined.

```
      <!-- HyTime Lexicographic Ordering Definition Notation -->
<![ %HyOrd; [
<!notation
   HyOrd          -- HyTime lexicographic ordering definition
                     notation --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Lexicographic Ordering Definition Notation
           (HyOrd)//EN"

-- CommonAttributes: altreps, included, superdcn --
>
]]><!-- HyOrd -->
```

A.2.4.1 HyTime lexicographic ordering definition notation (HyOrd)

The HyOrd notation provides a syntax for representing mappings from the members of lists of characters to the corresponding members of other lists of characters and from sequences of characters to other sequences of characters.

Characters lists for use in either kind of mapping are specified as SGML literals or as bracketed lists of decimal character numbers.

The syntax for HyOrd specifications is:

[26] HyOrd specification =
*HyOrd separator**,
HyOrd statement,
(*HyOrd separator+*,
HyOrd statement)*,
*HyOrd separator**,

[27] HyOrd separator =
s **|**
comment

[28] HyOrd statement =
correspondent character mapping **|**
character sequence mapping

[29] correspondent character mapping =
character list specification,
*HyOrd separator**,
pero,
*HyOrd separator**,
character list specification

[30] character sequence mapping =

character list specification,
*HyOrd separator**,
vi,
*HyOrd separator**,
character list specification

[31] character list specification =

literal character list |
numeric character list

[32] literal character list =

(*lit*,
*SGML character**,
lit) |
(*lita*,
*SGML character**,
lita)

[33] numeric character list =

dso,
*s**,
(*number*,
(*s+*,
number)*
*s**)?
dsc

The left-hand side of any HyOrd statement must be a list of at least one character. If several character sequences to be mapped begin with the same character sequence, only the longest sequence is mapped.

The right-hand side of a correspondent character mapping must be a list containing the same number of characters as the left-hand side of the statement, except that the right-hand side of a character sequence mapping may be an empty list of characters.

NOTE 391 The following example of a HyOrd specification illustrates one-to-none, one-to-one, one-to-many and many-to-one reorderings.

```
"adh" % [1 4 8]  -- a compares as 1, d compares as 4, and h compares as 8 --
"j"   = [4 27 8] -- j compares as dzh --
"l"   = [12]     -- l compares as 12 --
"ll"  = [13]     -- ll occurs between l and m --
"m"   = [14]     -- m compares as 14
"'"   = ""       -- apostrophe ignored in compare --
"z"   = [27]     -- z compares as 27 --
```

A.3 Architectural Form Definition Requirements (AFDR)

This annex states the requirements for the formal definition of the architectural forms by which an enabling document architecture governs the SGML representation of its documents.

NOTE 392 The formal definitions of the HyTime architectural forms satisfy these requirements and (without loss of generality) are used to illustrate them.

A.3.1 Enabling architectures

A document architecture, as that term is defined in ISO 8879, can be “encompassing”, governing every aspect of its documents' representation and processing. The document representation requirements for an encompassing architecture are expressed formally — at least insofar as SGML is capable of expressing them — in a document type definition (DTD).

A document architecture can also be “enabling”, in which case it does not specify complete document types. Instead, an enabling architecture defines rules, known as “architectural forms”, that application designers can apply in their document type definitions. These rules, and the associated architectural semantics, are described in an “architecture definition document”. The set of formal SGML specifications of the architectural forms, and related declarations for an enabling architecture, comprises a “meta-DTD”.

Conceptually, there are two steps to architectural processing. In the first step, generic architectural processing, a generic architecture engine validates a client document against the meta-DTD of its base architectures and, optionally, creates an architectural instance for each base architecture. In the second step, an architecture-specific semantic engine processes both the relevant architectural instances and the client document in order to implement and/or validate architecture-specific semantics.

NOTE 393 In practice, these steps may be combined into a single architecture- or application-specific processor. In other words, an application-specific processor may implement architecture-defined semantics without doing generalized architectural processing.

A.3.1.1 Architectural forms

Architectural forms are rules for creating and processing components of documents, just as document architectures are rules for creating and processing documents. There are four kinds:

element form This is defined by an element type declaration in conjunction with an attribute definition list declaration. The element type declaration can have a content model that constrains the elements conforming to the form.

attribute form This is defined solely by an attribute definition list declaration. Its attribute definitions can be used only with designated element forms.

notation form This is defined by a notation declaration in conjunction with an attribute definition list declaration.

data attribute form
This is defined solely by an attribute definition list declaration for a notation. Its attribute definitions can be used only with data entities conforming to the associated notations.

An element type whose instances conform to an element form defined by, say, the “ArcName” architecture, is informally called an “ArcName element type”, even though the element type itself is not defined by the ArcName architecture, but by an application. An instance of such an ArcName element type is called an “ArcName element”. An element, notation, data portion, or attribute that does not conform to an architectural form is termed “non-architectural”.

An application DTD can include element types of different enabling architectures, element types that conform to no formally-defined architecture (other than the encompassing architecture represented by the DTD itself), and element types that conform to several enabling architectures at once. Where multiple architectures govern a document, the rules of each are enforced without regard to objects that are non-architectural with respect to it.

The enabling architectures that govern a DTD are called its “base” architectures. The DTD is their client and its architecture is said to be “derived” from the base architecture. An enabling architecture can itself be derived from one or more base architectures.

NOTE 394 For example, the full HyTime architecture is derived from the HyBase architecture, which is in turn derived from the General Architecture.

An individual element cannot conform to more than one element type form of a given architecture. It can, however, conform to multiple attribute forms unless prohibited by the rules of the particular architecture.

NOTE 395 A derived architecture is unaware of whether its base architecture is itself a derived architecture. For this reason, an element can conform to the element forms of two base architectures even when one of the architectures is derived from the other.

Most architectural information is conveyed by attributes, which are classified as follows:

architectural attributes

These describe the semantics of the architecture. They are semantically defined in the architectural definition document, declared in the meta-DTD and client DTD or LPD, and specified in the client document instance or LPD.

architecture support attributes

These describe properties of a client document's use of the architecture as a whole, including the names of architecture control attributes (see below). Generic support attributes for all architectures are semantically defined in the AFDR, which provides templates for them. Architecture-specific ones are semantically defined in the architecture definition document, which should also provide templates for them. Support attributes are declared in the client DTD or LPD and specified using the default value parameter of the definitions.

architecture control attributes

These allow architectural parsing and processing to be controlled from the document instance. They are semantically defined in the AFDR, which provides templates for them. Control attributes are declared in the client DTD or LPD, and specified in the client document instance or LPD.

A.3.1.2 Architectural document

For an SGML document and each of its base architectures there exists an “architectural document” that is a derivation of the SGML document. The architectural document for a base architecture consists only of the element types, notations, attributes, and data defined in that base architecture's meta-DTD, as affected by the document's architecture control attributes.

A.3.2 SGML conventions

The syntax and structure of architectural forms and element types are specified rigorously using SGML markup declarations. The declarations conform to valid SGML syntax in every respect, except for two modifications to attribute definition list declaration syntax, as identified later in this annex.

NOTE 396 This practice permits the declarations to be copied for use as models when creating DTDs derived from the architecture. It also allows any DTD to serve as a meta-DTD.

NOTE 397 Although any DTD can serve as a meta-DTD, architecture definition documents (such as this International Standard) use a convention to distinguish meta-DTDs from application DTDs when both are being discussed. Declarations used to define architectures are presented with the declaration names in lower-case. Those that define application document types follow the normal SGML convention; their declaration names are presented in upper-case. Declarations that are templates (that is, they are intended to be customized and used in an application DTD) have their declaration names in mixed-case.

Some templates are actually “template forms”, in that they must first be customized into templates by an architecture definition document, then customized again for use in an application DTD. These declarations are presented as templates when they are in a state that requires customization prior to use in an application DTD, but they are presented as application declarations when further customization is merely optional (that is, when they can be used as-is if the user accepts the recommended names and attribute values).

A.3.2.1 Element forms

An element form is defined by an element type declaration, optionally in conjunction with an attribute list declaration.

A.3.2.1.1 Element type declaration

The structure of an element form is defined by a “meta” content model in the element type declaration. The model is “meta” in the sense that each “element type” in the content model is really an architectural form and means “any elements conforming to this form”. An application DTD could therefore define several element types conforming to a given architectural form. Moreover, each could have a different content model as long as it allowed content that satisfies the meta-model.

The text of the document that specifies the architecture (the “architecture definition document”) states whether the architecture associates any semantics with the content of an element form. If the text is silent, subelements have no special semantics in their role as content of their containing element. However, they still have their normal architectural semantics as individual elements.

NOTE 398 For example, no special HyTime semantics are defined for the content of an activity policy association rule element, but a contextual link element occurring in the content would retain its normal HyTime semantics.

As with a normal content model, a meta-model can incorporate inclusion exceptions. However, the meaning and effect are different in the two cases. In a normal content model, an inclusion means that an occurrence of an included element is transparent to its surroundings, which affects record-end handling in SGML and can have semantic consequences in architectures and applications. In a meta-model, inclusions are strictly a syntactic convenience with no implications for the included element.

Use of inclusions in an element type declaration's meta-model means that the architecture allows elements of the named forms to occur anywhere in the declared element's content in the document instance, including the content of subordinate elements. The application designer can choose, in his DTD, to require the elements to be proper subelements, inclusions, or both. The fact that an element type was an inclusion in the architecture does not necessarily mean it would be an appropriate inclusion in the DTD.

NOTE 399 An example of a content meta-model with an inclusion is:

```
<!element
   drydoc         -- Dull document element --

   - O (a|b|c)* +(x|y)
>
```

The construct “(a|b|c)*” is a model group while “+(x|y)” is called an “inclusion exception” in SGML. It means that the element types in parentheses can occur at any level of the element structure. In contrast, element types in the model group can occur only as “proper subelements”, in the immediate content.

The omitted tag minimization parameter is primarily a suggestion to the DTD designer. It may also affect architectural processing during the creation of architectural documents by enabling the recognition of architectural elements by normal SGML markup minimization rules. (For example, architectural data in a client document may imply the start of the architectural element that contains the data in the architectural document.)

A.3.2.1.2 Meta-DTD

The set of declarations defined in an architecture definition collectively represent a meta-DTD. The occurrence of an architectural object in a document instance is checked against the meta-DTD to see if it is allowed. Other objects are not checked.

Data is considered architectural only when it occurs at a point where the meta-DTD allows data. This rule includes data in the content of non-architectural elements (unless architectural processing has been suppressed for the descendants of that element as described in *A.3.5.3 Architecture suppressor attribute*). Otherwise, data does not affect the checking against the meta-DTD.

NOTE 400 This requirement means that architectural form definitions need not provide for the occurrence of non-architecture element types. For example, in HyTime, the following would be valid, even though the nameloc form does not include title or description subelements:

```
<!ELEMENT MyNameLoc
  - O (Title?,nmlist+,Desc?)
>
<!ATTLIST MyNameLoc
  HyTime   NAME     #FIXED nameloc
  ...
>

...

<mynameloc>
<title>All About Otters</title>
<nmlist>aao</>
<desc>
Locates the section on "Athletic Achievements"
</desc>
</mynameloc>
```

Any failure of a document instance to conform to the meta-DTD is a reportable architecture error (RAE). In particular:

- It is an RAE if an element occurs in a document instance at a point where its architectural form is not permitted by the meta-DTD of that form's architecture.
- It is an RAE if, at a point in a document instance where a meta-DTD requires data or an instance of a particular architectural form, the required object fails to occur.
- It is an RAE if the value of an architectural attribute does not satisfy its declared value.

An architecture does not constrain the construction of document type definitions, only document instances. For example, any document's DTD can declare HyTime as a base architecture, as long as its document instance conforms to HyTime.

NOTE 401 A validating architecture engine can optionally report DTD constructs (such as poorly designed content models) that would prevent the creation of a valid conforming instance, or that would allow the creation of a nonconforming instance.

A meta-DTD is flexible and does not necessarily clarify all issues of contextual relevance; some elements may be meaningful only in the context of other elements from which they are referenced, or only as resources for the processing of other elements.

The meta-DTD comprising all of the SGML definitions of an architecture necessarily includes all of the architecture's optional facilities (if any). In contrast, for any given architecture implementation's system declaration (see *A.3.4 Architecture support declarations*) there will be a single meta-DTD that permits only the optional facilities supported by that system. Moreover, the actual meta-DTD applicable to a given document could be more restrictive than either of those, depending on the facilities required by that document (see *A.3.6.1 Architectural document element*). The constraints of a more restrictive meta-DTD that is required by a document's architecture support declarations can also be enforced by the document's DTD.

NOTE 402 The declared content or content model of an application's element type can differ from the meta-model of the architectural form as long as it allows instances that conform to the architectural form meta-model. For example, a meta-model might be “(a|b|c)”. An application designer could legally exclude the possibility of “c” by using the content model “(a|b)”. The content model “(a|x)” could also be used. In that case, if “x” is not architectural (it is not derived from any form in the architecture or is not taken to be architectural) it would not be an architectural error for “x” to occur (because non-architectural elements are simply ignored for the purpose of validation against an architectural meta-DTD). If “x” is architectural, it would be an RAE.

As with normal DTDs, parameter entities can be used in the construction of meta-DTDs. The parameter entity declarations are primarily an aid for tailoring the meta-DTD to correspond to any modularization of facilities supported by the architecture definition document. They can be used in conjunction with marked sections for automatic tailoring in conjunction with the ArcOptSA attribute (see *A.3.5 Architecture control attributes*). They can also serve to document relationships among the architectural forms, but otherwise have no semantic significance.

The meta-DTD of a derived enabling architecture is a “derived meta-DTD”. The meta-DTDs of its base architectures are its “base meta-DTDs”. The meta-DTDs of base architectures of application DTDs are the application's base meta-DTDs.

A.3.2.1.3 Attribute definition list declaration

An architectural element type's attribute definition list in an application DTD includes the ArcForm attribute, which identifies the architectural form. The ArcForm attribute is unique in that it must be defined for every element type whose instances are to be processed in a manner defined by the architecture (unless architectural markup minimization is used, as described in *A.3.6.2 Architectural markup minimization*).

NOTE 403 Normally, every element of the same type has the same architectural form, which allows the ArcForm value to be fixed in the DTD. However, this is not a requirement: an element type definition, could, for example, allow its instances to be one of a choice of architectural forms.

The other attribute definitions follow the conventions described in *A.3.2.3 Attribute list conventions*.

A.3.2.2 Attribute forms

An attribute form defines attributes that can be used in the attribute definition lists of element types conforming to one or more element forms, including SGML processing link attribute lists. The element forms with which the attributes can be used are indicated by declaring their names as associated element types.

In a meta-DTD, unlike a normal DTD, there can be more than one attribute definition list declaration associated with the same element type. Moreover, if the associated element type is specified as the reserved name “#ALL”, the attributes can be used with any of the architecture's element forms. (Such attributes are known as “common attributes”.) These two facilities are part of the “AFDR meta-DTD notation” extensions and can be used only in a meta-DTD.

It is not an error if the same attribute name occurs in more than one such declaration, but later definitions of the same attribute name for the same associated element type have no effect. The order in which the declarations are parsed governs, except that common attribute declarations are considered to follow all the other attribute definition list declarations.

NOTE 404 As SGML requires that an attribute name be unique in its attribute list, designers should make sure that a common attribute — which could occur in all attribute lists — has a name that is different from all other attribute names in the architecture.

NOTE 405 In an application DTD, “#ALL” cannot be specified as SGML does not allow it. Nor can more than one attribute definition list declaration be associated with the same element type.

An attribute form can also be defined for use with notation forms by specifying their names as associated notation names in the attribute definition list declaration. In meta-DTDs, the same extensions are available for these as for associated element types. The same rules regarding common data attributes can be used with all of the architecture's data attribute forms.

A.3.2.3 Attribute list conventions

All architectural attributes, whether part of an element form or an attribute form, are specified using the conventions in this sub-clause.

NOTE 406 For a given element or notation form, an application can replace an attribute's architecture-defined name by using the “ArcNames” attribute (see *A.3.4.2 Architecture support attributes*).

It is a reportable architecture error if a document instance contains a value for an attribute that fails to satisfy the declared value defined for it in the meta-DTD.

NOTE 407 As an attribute value must satisfy an attribute definition in both the meta-DTD and the client DTD, there is a possibility for a conflict of concrete syntaxes. For this reason, it is recommended that the concrete syntax of a meta-DTD be designed for use with all expected client documents. The broader the potential use, the closer the concrete syntax (and particularly its name spelling rules) should be to the reference concrete syntax (or the usual concrete syntax of the expected user community, if different).

A.3.2.3.1 Default value prescription

An architecture's attributes are categorized as being “mandatory” or “non-mandatory”, and also as being “constant” or “variable”, as follows:

— Non-mandatory attributes

These attributes may be defined in the application DTD, but need not be. If an attribute is not defined in the client DTD, the architecture either provides a default value either as part of the attribute definition in the architecture meta-DTD, or, in the case of an impliable or content reference attribute, documents the default behavior as part of the architecture definition document. The non-mandatory status of an attribute is documented in the default value prescription of the attribute definition by the presence of “#IMPLIED”, “#CONREF”, or a literal (possibly fixed) default value.

If a non-mandatory attribute is not defined, or a defined impliable attribute has no default value and is not specified on the start-tag, the architectural semantic processor uses the default value documented in the architecture definition document.

NOTE 408 An application can conveniently specify the defaulting of impliable attributes by using the default value list facility of the General Architecture (see *A.5.6 Default value list*).

— Mandatory attributes

These attributes must be defined in the application DTD and values must be provided by the application, either by defaulting or by an attribute specification. The mandatory status of an attribute is documented in the default value prescription of the attribute definition by the presence of “#REQUIRED”.

When an architecture definition document modularizes an architecture, an optional facility could have attributes that are mandatory with respect to that facility. The meta-DTD used by a client document must contain all the declarations that are required by the facilities used by the document. It need not contain declarations that are required by optional facilities that are not used.

— Constant attributes

The default value prescription of an architectural attribute may be the token “#FIXED” followed by a default value specification; this denotes a constant attribute whose value must always be equal to its default.

NOTE 409 If an attribute is fixed in the meta-DTD, an application should not declare it at all; this ensures that it will always be assigned its default value. Alternatively, it may be declared as a fixed attribute with a default value identical to the architectural default value.

— Variable attributes

An attribute that is not a fixed attribute is a variable attribute.

In an application DTD, the default value prescription for a variable attribute, whether mandatory or not, can use any keyword or attribute value specification allowed by SGML. The form used to document the attribute in the architecture definition is irrelevant.

NOTE 410 Some attributes defined in the HyTime architecture are variable with respect to their associated element forms, but are required to be constant with respect to client element types. See *5 Notation* for details.

NOTE 411 The following list gives examples of the forms of declared value and default value prescriptions used to document each combination of attribute categories:

— Mandatory

```
NAMES     #REQUIRED
```

— Non-mandatory/constant

```
NOTATION  #FIXED SDQL
```

— Non-mandatory/variable

```
NUMBER    #IMPLIED
NAME      #CONREF
NAMES     novamp
(order|disorder) order
CDATA     ""
```

A.3.2.4 Processing link attributes

SGML processing link attribute lists can be used to specify both architectural attributes and architecture control attributes for base architectures declared in the source document DTD. When an element has both a link attribute and an SGML source element attribute with the same name, only the link attribute will be considered for architectural processing, with the exception of the “ArcNames” architecture control attribute (see *A.3.5.2 Architectural attribute renamer*).

A.3.3 Architecture base declaration

A document indicates conformance to architectures by using an architecture base declaration (ArcBase).

An ArcBase declaration identifies one or more base architectures of a DTD or a derived meta-DTD. It cannot be used in a meta-DTD that is not derived. It should precede the architecture support declarations pertaining to the base architectures that it identifies. There can be more than one ArcBase declaration in a DTD or meta-DTD.

Syntactically, the ArcBase declaration is a processing instruction (PI), not an SGML markup declaration. In the template below, it is shown in the reference concrete syntax. In use, the ArcName-list parameter must be replaced by one or more architecture names (ArcName) declared as notation names, separated by SGML *s* separators (white space). It is an RAE if the DTD or meta-DTD does not declare a notation for each ArcName specified..

The declaration name is the character string following the initial keyword “IS10744” and one or more s separators, up to the first s separator. The declaration name is subject to upper-case substitution if the SGML declaration of the client document specifies general upper-case substitution.

The name is normally “ArcBase” but it can be changed using the APPINFO parameter of the SGML declaration, if necessary, to avoid the (admittedly unlikely) possibility of conflicts when retrofitting an architecture to a document that already has PIs that begin with “IS10744 ArcBase ” (see *A.3.3.1 Enabling architecture use of APPINFO parameter*).

The ArcBase declaration will only be recognized in the DTDs and LPDs with respect to which the document is being parsed.

```
               <!-- Architecture Base Declaration -->
        <!-- TEMPLATE FOR PI IN DTD OR DERIVED META-DTD -->
<?IS10744 ArcBase ArcName-list >
```

A.3.3.1 Enabling architecture use of APPINFO parameter

To determine when enabling architectures are in use, the APPINFO parameter of the SGML declaration is parsed as a space-delimited sequence of tokens. Conformance of a document to one or more architectures defined in accordance with these requirements is indicated by specifying the keyword “ArcBase” as such a token. The keyword indicates the potential presence in one or more DTDs or LPDs of an architecture base (ArcBase) declaration that identifies the “architecture support declarations” for the architectures on which the DTD is based. The use of the APPINFO parameter to indicate conformance to an architecture is optional.

NOTE 412 The use of APPINFO serves to declare the use of architectures within SGML declarations. However, it is sufficient to enable architectural processing to declare the use of an architecture with the ArcBase declaration within DOCTYPE declarations.

The format of the token is:

```
ArcBase
```

where “ArcBase” is not case-sensitive. The token can also specify the name of the ArcBase declarations in the document's DTDs and LPDs if it is other than “ArcBase”. The format is:

```
ArcBase ArcBase="NewBase"
```

The APPINFO parameter is interpreted as an s-separator-delimited list of tokens.

Specification of the ArcBase name to be used in documents is optional when the name of the ArcBase declarations is ArcBase.

A.3.4 Architecture support declarations

A DTD contains architecture support declarations for each architecture on which its instances are based. There is a notation declaration that identifies the architecture definition document (including the architecture version, if any) and an external entity that contains the meta-DTD. Associated with the notation declaration is an attribute definition list declaration for “architecture support attributes”.

NOTE 413 It is not necessarily an error if the definition document and/or meta-DTD cannot be accessed, as an implementation might not require access to these objects. The primary purpose of the declarations is to declare and specify architecture support attributes. However, the meta-DTD must be accessible if the architecture engine is to produce a parsed architectural document.

To the extent permitted by SGML, architecture support declarations should precede other declarations.

A.3.4.1 Architecture notation declaration

Each base architecture name (ArcName) specified in an ArcBase declaration must be declared as a notation name in a notation declaration that identifies the architecture definition document. (It is recommended that such notation declarations be identified by the use of a formal public identifier in which the public text description component begins with the words “AFDR ARCBASE”.)

The template for such a declaration is shown below. In use, ArcName is replaced by the actual notation name for the architecture, as specified in the ArcBase declaration. The conformance statement shows that the architecture client claims conformance to the AFDR. (It does not indicate conformance to the HyTime architecture.)

```
              <!-- Architecture Notation Declaration -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Notation
   ArcName        -- ArcName architecture --
                  -- A base architecture used in conformance with the
                     Architectural Form Definition Requirements of
                     International Standard ISO/IEC 10744. --

   PUBLIC ".../NOTATION AFDR ARCBASE ArcName Architecture//.."
                  -- Constraint: Public ID of ArcName architecture
                     definition document --
>
```

A.3.4.2 Architecture support attributes

There must be an attribute definition list declaration for each architecture notation declaration to define mandatory architecture support attributes. An individual architecture can define additional architecture support attributes of its own.

The template for such a declaration is shown below. In use, ArcName is replaced by the actual notation name for the architecture and the default values prescriptions are replaced by the desired values of the support attributes. For non-mandatory attributes, the “Default:” conventional comment states the effect of not declaring the attribute.

The attribute **architectural form attribute name** (*ArcFormA*) specifies the name of an architecture control attribute (hereafter identified as the “ArcForm” attribute) that the client DTD must define for every architectural element type and notation to identify its architectural form (see *A.3.5.1 Architectural form attribute*).

NOTE 414 This attribute does not appear in the meta-DTD unless the meta-DTD is also a client DTD, in which case the attribute's value would be an architectural form name from its base architecture, not from the client DTD. For example, an architecture derived from HyTime would define its own architecture naming attribute, e.g. “MyArch”. This attribute would be used in client documents derived from the MyArch architecture but would not appear in the MyArch meta-DTD. Because the MyArch architecture is derived from HyTime, the meta-DTD would use the HyTime attribute to define the derivation of MyArch element forms from HyTime element forms.

NOTE 415 The ArcName can be used as the name of the ArcForm attribute. For example, “HyTime” is both an architecture name and the name of that architecture's ArcForm attribute.

The attribute **architectural attribute renamer name** (*ArcNamrA*) specifies the name of an architecture control attribute (the “ArcNames” attribute) that allows an application to substitute its own names for architectural attribute names (see *A.3.5.2 Architectural attribute renamer*).

The attribute **architecture suppressor attribute name** (*ArcSuprA*) specifies the name of an architecture control attribute (the “ArcSupr” attribute) that controls suppression of architectural processing (see *A.3.5.3 Architecture suppressor attribute*).

The attribute **architecture ignore data attribute name** (*ArcIgnDA*) specifies the name of an architecture control attribute (the “ArcIgnDA” attribute) that controls whether the data content of an element is treated as architectural.

The attribute **architecture document element form name** (*ArcDocF*) specifies the name of the document element form (the “ArcDoc” form) (see *A.3.6.1 Architectural document element*).

The attribute **architecture meta-DTD entity** (*ArcDTD*) identifies the external entity that contains the meta-DTD to which the architectural document instance conforms. The value of the attribute is either the name of a parameter entity prefixed with the pero delimiter or the name of a general entity. It must be subject to the same SGML declaration as the client document, except that its quantity set could differ. The difference is specified by the attribute **architecture quantity set** (*ArcQuant*), which is a list of quantity and value pairs as found in the quantity set parameter of an SGML declaration. The values for the quantities named in the attribute value apply to the meta-DTD and architectural instance in place of those in the client document SGML declaration. (See *A.3.4.3 Architecture entity declaration*). If no value is specified, then the name of the meta-DTD entity is taken to be the name of the architectural notation after the application of SGML name and entity case folding rules.

NOTE 416 In the reference syntax, this means that the meta-DTD entity name must be in upper case.

The following three attributes are used in connection with “architectural markup minimization” (which is described in *A.3.6.2 Architectural markup minimization*):

The attribute **architecture data form name** (*ArcDataF*) specifies the name of an architectural notation form (the “ArcData” form) that is used as the default for external data entities.

The attribute **architecture bridge form name** (*ArcBridF*) specifies the name of an architectural form (the “ArcBrid” form) that bridges between the architecture and a non-architectural element or notation. It is the default architectural form for elements that are architectural but do not have a defined architectural form. Such elements are usually included because they exhibit an ID attribute.

The attribute **architecture automatic form mapping** (*ArcAuto*) indicates whether to map element types and external data entity notations to identically named architectural forms (ArcAuto) or not (nArcAuto).

When an architecture definition document modularizes an architecture, an application designer can define additional support attributes that identify the facilities for which support may be required by the document. Additional support attributes can also be used to specify attributes that are properties of documents (as opposed to being properties of the document element), or to set parameters for the initialization of architectural processing. The architecture support attribute declaration is passed by the architecture engine to the architectural semantic processor at the end of the prolog.

One useful modularization technique, which was applied to the HyTime architecture, is to create a master meta-DTD that is modularized by means of marked sections. Each marked section start contains a parameter entity reference whose entity name is that of the facility (or optional component of a facility) whose declarations are in the marked section. When those entities are declared with the entity text “INCLUDE”, the declarations are included in the meta-DTD; when the entity text is “IGNORE”, they are not.

The attribute **architecture options support attribute names** (*ArcOptSA*) specifies the names of one or more architecture support attributes. The support attributes list the names of parameter entities whose entity text should be declared as “INCLUDE” in the meta-DTD. The default architecture support attribute is **architecture options** (*ArcOpt*).

NOTE 417 A single architecture options attribute will suffice for simple architectures, which may have little or no modularization. For a large modularized architecture such as HyTime, it can be convenient to have a separate options attribute for each module that names the included optional facilities of that module.

It is an error if a document requires an architecture facility that is not identified in the support declarations, but a system is not required to report such an error as it may not be able to do so. However, it is not an RAE if the declarations identify facilities that are not actually required by the document.

NOTE 418 A system is free to supply facilities for which support requirements were not declared, if it is capable of doing so.

If a system attempts to process a document whose support declarations identify facilities that the system does not support (see *A.3.4 Architecture support declarations*), the system must report an error even if the document does not actually use those facilities.

NOTE 419 In other words, a system can give the user the option of ignoring an “unsupported facility” error, but it must report the error in the first instance.

Identification of a facility implies automatic identification of any facilities on which it is dependent.

```
               <!-- Architecture Support Attributes -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Attlist #NOTATION
   ArcName        -- Base architecture --

-- Support attributes for all architectures --

   ArcFormA       -- Architectural form attribute name --
      NAME        -- Constraint: must be unique among all attribute
                     names in DTD in which it is specified --
      #IMPLIED    -- Default: ArcName --

   ArcNamrA       -- Architectural attribute renamer attribute name --
      NAME        -- Constraint: must be unique among all attribute
                     names in DTD in which it is specified --
      #IMPLIED    -- Default: no renaming --

   ArcSuprA       -- Architecture suppressor attribute name --
      NAME        -- Constraint: must be unique among all attribute
                     names in DTD in which it is specified --
      #IMPLIED    -- Default: no suppression --

   ArcIgnDA       -- Architecture ignore data attribute name --
      NAME        -- Constraint: must be unique among all attribute
                     names in DTD in which it is specified --
      #IMPLIED    -- Default: data is conditionally ignored --

   ArcDocF        -- Architecture document element form name --
      NAME        -- Constraint: name of element form in architecture
                     meta-DTD --
      #IMPLIED    -- Default: ArcName --

   ArcDTD         -- Architecture meta-DTD entity --
      CDATA       -- Lextype: (ENTITY|PENTITY) --
      #IMPLIED    -- Default: entity whose name matches the notation name
                     after SGML name and entity folding is applied. --
                  -- Constraint: a name must be specified or the meta-DTD
                     entity name after folding must match notation name
                     after folding. --

   ArcQuant       -- Architecture quantity set --
      CDATA       -- Lextype: (NAME,NUMBER)+ --
```

```
                 -- Constraint: quantity name/value pairs --
      #IMPLIED    -- Default: no variation --

   ArcDataF       -- Architecture data form name --
                  -- For external data entities whose notation names
                     don't match architectural notations --
      NAME        -- Constraint: name of notation form in architecture
                     meta-DTD --
      #IMPLIED    -- Default: no defaulting --

   ArcBridF       -- Architecture bridge form name --
                  -- For elements with an ID and no ArcForm attribute
                     whose form isn't defaulted --
      NAME        -- Constraint: name of element form in architecture
                     meta-DTD --
      #IMPLIED    -- Default: no defaulting --

   ArcAuto        -- Architecture automatic form mapping --
                  -- Automatic form mapping for identically named
                     element types and external data entity
                     notations --
      (ArcAuto|nArcAuto)
      ArcAuto

   ArcOptSA       -- Architecture options support attribute names --
      NAMES       -- Lextype: ATTNAME+ --
      ArcOpt

-- Support attributes for this architecture --

   ArcOpt         -- Architecture options --
      CDATA       -- Lextype: csname+ --
                  -- Constraint: parameter entities in meta-DTD to be
                     set to "INCLUDE" --
      #IMPLIED    -- Default: none --

-- Other support attributes for this architecture may be added by
   meta-DTD designer. --
>
```

A.3.4.3 Architecture entity declaration

The value of the ArcDTD support attribute must be declared as an entity name in an entity declaration that identifies an architecture meta-DTD to which the document instance conforms. The meta-DTD entity must conform to the SGML declaration of the client document, except for the quantity set variations specified in the ArcQuant support attribute.

If the architecture has architecture-specific support attributes, it is not an error if they are more or less restrictive than the meta-DTD. The document instance must conform to both.

It is possible (though not required) for meta-DTDs to be processed by a generic “architecture engine”. Such an engine behaves similarly to an SGML parser validating conformance to a DTD, except that:

a) Element form names, rather than GIs, are checked against the meta-content models.

b) The AFDR meta-DTD notation is recognized (see *A.3.2.2 Attribute forms*).

Several templates for architecture entity declarations are shown below. In use, the string “ArcName” is replaced by the actual notation name for the architecture. The choice of template depends on whether the AFDR meta-DTD notation is used and, if not, whether the meta-DTD is also incorporated within the client DTD by means of a parameter entity reference.

NOTE 420 In either case, it is only accessed directly by the architecture engine (if at all), outside the parsing context of the document.

```
              <!-- Architecture Entity Declarations -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
    <!-- For use when AFDRMeta extensions are used in meta-DTD -->
<!NOTATION
   AFDRMeta       -- AFDR Meta-DTD Notation --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD Notation//EN"
>
<!Entity
   ArcName        -- ArcName architecture meta-DTD --

   PUBLIC ".../DTD AFDR Meta-DTD ArcName Architecure//.."
                  -- Constraint: Public ID of ArcName architecture
                     meta-DTD --
   CDATA AFDRMeta
>

<!-- For use when AFDRMeta extensions are not used and meta-DTD is not
     referenced as part of DTD -->
<!NOTATION
   SGML           -- Standard Generalized Markup Language --

   PUBLIC "ISO 8879:1986//NOTATION
           Standard Generalized Markup Language//EN"
>
<!Entity
   ArcName        -- ArcName architecture meta-DTD --

   PUBLIC ".../DTD AFDR Meta-DTD ArcName Architecure//.."
                  -- Constraint: Public ID of ArcName architecture
                     meta-DTD --
   CDATA SGML
>

<!-- For use when AFDRMeta extensions are not used and meta-DTD is
     referenced as part of DTD -->
<!Entity %
   ArcName        -- ArcName architecture meta-DTD --

   PUBLIC ".../DTD AFDR Meta-DTD ArcName Architecure//.."
                  -- Constraint: Public ID of ArcName architecture
                     meta-DTD --
>
```

A.3.5 Architecture control attributes

Architecture control attributes allow architectural parsing and processing to be controlled from the client document instance. The names of the architecture control attributes are specified as values of corresponding architecture support attributes (see *A.3.4.2 Architecture support attributes*).

NOTE 421 Note that a single client element may have as many sets of architectural control attributes as the client document has base architectures.

An architecture does not declare architecture control attributes in a meta-DTD. Instead, the client declares them for those element types and notations that require them.

A.3.5.1 Architectural form attribute

The architecture control attribute **architectural form** (*ArcForm*) identifies the architectural form of an element or notation. Its value is the name of the architectural form to which the element or notation conforms.

The actual name of the attribute should not be ArcForm; it must be unique for each base architecture and should ideally be unique among all the attribute names in client documents. The actual name is specified by the ArcFormA attribute of the architecture notation declaration.

A client must declare the ArcForm attribute for every architectural element and external data entity, except where architectural markup minimization is used (see *A.3.6.2 Architectural markup minimization*).

NOTE 422 An architecture engine uses this attribute to determine the architecture-specific processing that the architectural semantic processor will perform, and to know which architecture-defined attributes to examine. The application processor supplements this processing by using the application-defined element type and other attributes to determine the application-specific processing to perform. In object-oriented programming terms, an application-defined derived class (the element type) inherits the properties of its base class (the architectural form).

It is an RAE if a document instance contains a value for the architectural form attribute other than one specified in the meta-DTD.

NOTE 423 In other words, an architecture's set of architectural forms cannot be extended by an application (but see *A.3.6.3 Derived enabling architectures*).

```
                <!-- Architectural Form Attribute -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Attlist
   ElemTypeName   -- Client element type --

   ArcForm        -- Base element form name --
      NAME        -- Constraint: name of element form defined in base
                     architecture's meta-DTD --
      #IMPLIED    -- Default: non-architectural unless found to be
                     architectural via automatic form mapping or
                     architectural bridging --
>
<!Attlist #NOTATION
   NotationName   -- Client notation --

   ArcForm        -- Base notation form name --
      NAME        -- Constraint: name of notation form defined in base
                     architecture's meta-DTD --
      #IMPLIED    -- Default: non-architectural unless found to be
                     architectural via automatic form mapping or
                     architectural bridging --
>
```

A.3.5.2 Architectural attribute renamer

The architecture control attribute **architectural attribute renamer** (*ArcNames*) is an attribute that defines client substitutes for architecture attribute names. The substitute is treated as if it were the correspondingly-named architecture-defined attribute or architectural content. An attribute bearing an architecture-defined attribute name for which a substitute is defined will be ignored by the architecture engine. If the substitute name is "#DEFAULT", then an attribute bearing the architecture-defined attribute name is ignored in the usual way, but the value of the attribute is not taken from another attribute but is instead defaulted as specified in the meta-DTD.

If the substitute name is "#CONTENT", the value of the attribute occurs as the syntactic content of the element in the client document. The syntactic content of the element is converted into the attribute value by concatenating the data from the leaves of the tree rooted at that element. In this case the syntactic content is not treated as architectural content.

NOTE 424 The architectural content will therefore be empty unless some attribute value in the client document is serving as the source of architectural content because of the use of "#ARCCONT".

A substitute name may be followed by one or more triples of tokens each consisting of "#MAPTOKEN" followed by two name tokens. These specify user substitutes for architecture-defined tokens occurring in the value of this attribute. If a token in a value specified for this attribute in the client document is equal to the second token, then it will be replaced by the first token. If "#MAPTOKEN" is specified for an architecture-defined attribute whose declared value is CDATA then the value of the attribute shall be tokenized before tokens are substituted.

NOTE 425 The attribute might have to be declared as CDATA because, for example, it could contain tokens starting with the RNI delimiter.

Architectural syntactic data content can be entered as an attribute value by specifying the architectural attribute name as "#ARCCONT" and naming the client attribute in whose value it will occur. In this case, any syntactic content in the client document is considered non-architectural.

NOTE 426 As always, the data must satisfy the lexical and semantic constraints of both the architectural form and the application element type. For example, although comment declarations can be intermixed with parsed character data in syntactic content, they are not allowed if the data is entered as an attribute value.

A single element may have both an ArcNames source element attribute and an ArcNames link attribute. In an ArcNames source element attribute value, an ATTNAME must refer to a source element attribute. In an ArcNames link attribute value, an ATTNAME is first assumed to be that of a link attribute. However, if no such link attribute exists, an identically named source element attribute will be used if it exists.

It is an RAE if ArcNames is used in a way that causes conflicts or duplications.

```
               <!-- Architectural Attribute Renamer -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Attlist
   ElemTypeName   -- Client element type --

   ArcNames       -- Architectural attribute renamer --
                  -- Defines user names for architecture attributes --
      CDATA       -- Lextype: ((NAME,(ATTORCON|"#DEFAULT"),
                                ("#MAPTOKEN", NMTOKEN, NMTOKEN)*)|
                               ("#ARCCONT",ATTNAME))* --
                  -- Constraint: a given ATTNAME or NAME, #CONTENT, or
                     #ARCCONT can occur only once --
                  -- Constraint: architecture name precedes user
```

```
                    name --
      #IMPLIED    -- Constant --
                  -- Default: no renaming --
>
<!Attlist #NOTATION
   NotationName   -- Client notation --

   ArcNames       -- Architectural attribute renamer --
                  -- Defines user names for architecture attributes --
      CDATA       -- Lextype: (NAME,(ATTNAME|"#DEFAULT"),
                              ("#MAPTOKEN", NMTOKEN, NMTOKEN)*)* --
                  -- Constraint: a given ATTNAME or NAME can occur
                     only once --
                  -- Constraint: architecture name precedes user
                     name --
      #IMPLIED    -- Default: no renaming --
>
```

A.3.5.3 Architecture suppressor attribute

The architecture control attribute **architecture suppressor** (*ArcSupr*) suppresses or restores architectural processing for the descendants of an element.

sArcAll — Completely suppress all architectural processing of descendants. It is not possible to restore architectural processing for a descendant.

sArcForm — Suppress processing of the ArcForm attribute of all descendants of this element, except for those elements that have a non-implied ArcSupr attribute.

sArcNone — Don't suppress architectural processing for the descendants of this element.

The value may also be implied, in which case the state of architectural processing is inherited.

If an element has an ArcSupr attribute that was processed, its ArcForm attribute will always be processed. Otherwise its ArcForm attribute will be processed unless its closest ancestor that has a non-implied value for the ArcSupr attribute suppressed processing of the ArcForm attribute. An element whose ArcForm attribute is processed will not be treated as architectural if it has an implied value for the ArcForm attribute.

NOTE 427 For example, “sArcAll” suppresses all architectural processing at any level of the element's content, while “sArcForm” allows the possibility that a descendant element's ArcSupr attribute could restore processing within its own content.

NOTE 428 The suppression of architectural processing can be useful if architectural elements are to be used for other purposes; for example, to display or edit HyTime resource elements within a HyTime document.

NOTE 429 When sArcForm is specified, meta-model checking is suspended. The element can therefore contain any of an architecture's elements, even those that normally are not part of the ArcCFC (that is, that can occur only in the content of some other architectural element). This freedom is possible because the architectural processing is suppressed, and therefore does not require the processing context that would have been established by the omitted containing element.

```
              <!-- Architecture Suppressor Attribute -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Attlist
   ElemTypeName   -- Client element type --

   ArcSupr        -- Architecture suppressor --
                  -- Suppress architectural processing --
```

```
      (sArcAll|sArcForm|sArcNone)
      #IMPLIED    -- Default: inherited --
>
```

A.3.5.4 Architecture ignore data attribute

The architecture control attribute **architecture ignore data** (*ArcIgnDA*) controls whether the data content of an element is treated as architectural.

nArcIgnD Data is not ignored. It is an error if data occurs where not allowed by the meta-DTD.

cArcIgnD Data is conditionally ignored. Data will be ignored only when it occurs where the meta-DTD does not allow it.

ArcIgnD Data is always ignored.

The value may also be implied, in which case the state of architectural processing is inherited. If the document element has no value specified, cArcIgnD will be used.

```
             <!-- Architecture Ignore Data Attribute -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Attlist
   ElemTypeName   -- Element type name --

   ArcIgnDA       -- Architecture ignore data --
      (nArcIgnD|cArcIgnD|ArcIgnD)
      #IMPLIED    -- Default: inherited from parent element; document
                     element defaults to cArcIgnD --
>
```

A.3.6 Other architecture-related considerations

This clause discusses other considerations related to architectures.

A.3.6.1 Architectural document element

Every architecture must provide the element form **architectural document element** (*ArcDoc*), to which the document element of a client document must conform if it does not already conform to another element form in the architecture. The actual name of the architectural form need not be ArcDoc; it could reflect the name of the architecture. The actual name is specified by the ArcDocF attribute of the architecture notation declaration.

The ArcDoc form allows an architecture to define attributes for the document element, and to specify the “architectural context-free content” (ArcCFC). The ArcCFC is the set of element forms that can occur independently at the highest level of a document.

NOTE 430 As not all architectural elements have constrained content models, there could be many forms for which ArcCFC is the permitted content. A useful convention is to declare an ArcCFC parameter entity for reference in element form declarations.

A template for declaring a document element form is shown below. In use, ArcDoc is replaced by its actual name and a name other than ArcCFC could be used for the content entity. The architecture designer could add the architecture's own attribute definitions for the document element form.

NOTE 431 Designers may choose to have meta-inclusions in %ArcCFC as well as a model group. Alternatively, they may choose not to have an ArcCFC entity at all.

```
               <!-- Architectural Document Element -->
     <!-- TEMPLATE FOR DOCUMENT ELEMENT FORM IN ANY META-DTD -->
<!Entity %
   ArcCFC         -- Architectural context-free content --

   "ANY"
>
<!Element
   ArcDoc         -- Architectural document element --

   - O
   %ArcCFC;
>
<!Attlist
   ArcDoc         -- Architectural document element --

-- Document attributes for this architecture may be added by meta-DTD
   designer. --
>
```

A.3.6.2 Architectural markup minimization

Architectural markup minimization allows the determination of the architectural form of an element or an external data entity even when there is no explicitly specified ArcForm attribute. The value of the ArcAuto architecture support attribute determines whether minimization is in effect. If so, the values of the ArcBridF and ArcDataF architecture support attributes determine the default form, in accordance with the following procedure:

1) If the ArcForm architecture control attribute is declared:

   a) If ArcForm is implied, the object is non-architectural.

   b) If ArcForm has a value, the architectural form is that value.

2) If the ArcForm architecture control attribute is not declared:

   a) If the object is the document element, the architectural form is ArcDoc.

      NOTE 432 This is a mandatory form of minimization; the others are optional.

   b) If not, and the ArcAuto support attribute is ArcAuto and the element type or notation name of the object is the same as that of an element form or notation form name in the meta-DTD, that form is considered the architectural form of the object.

      NOTE 433 The external identifiers in the meta- and client notation declarations need not be the same.

   c) If not, and the object is an element with an ID, use the ArcBrid form name if one was specified.

   d) If not, and the object is an external data entity, use the ArcData form name if one was specified.

   e) If not, the object is not architectural.

A.3.6.3 Derived enabling architectures

An enabling architecture can be derived from one or more base architectures by the same means that a DTD is derived from one or more base architectures. That is, an ArcBase declaration and support declarations for the specified base architectures are present in the meta-DTD. A derived architecture can also serve as a base architecture for DTDs and for other derived architectures.

NOTE 434 This facility provides designers the tools for subclassing and scaling SGML information bases without compromising the accuracy and consistency obtained from SGML's rigorous document type conformance checking. It brings to document data modeling OOPS benefits like multiple inheritance and encapsulation, while preserving SGML's ability to control a document's properties through a single coherent document type definition.

NOTE 435 A conforming instance of a derived architecture provides the information needed to construct a conforming instance of each of its base architectures. Therefore, if a system does not support the semantics of a derived architecture, it could, as an error-recovery fallback, treat the document as an instance of a base architecture instead.

Just as an architecture engine can construct an architectural document for a base architecture of a client document, so can it construct architectural documents for the base architecture(s) of architectural documents.

A.3.6.4 Relating applications and architectures

The following rules apply with respect to the relationship of attributes declared for an element form in a meta-DTD (“base attributes”) and attributes of an element of that form in a client document (“client attributes”):

— If a base attribute has a declared value prescription of ID (an “ID attribute”), then a client ID attribute will be treated as the corresponding architectural attribute regardless of whether the attribute names are the same or whether there is an ArcNames attribute.

— If a value is specified for a client attribute whose base attribute is an ID attribute, the client attribute must also be an ID attribute.

— A client attribute value satisfies a base attribute whose declared value is ENTITY or ENTITIES if it is the name of an architectural general entity declared in the client document. An entity is architectural if it is a SUBDOC or an external data entity that has an architectural form.

  NOTE 436 An external data entity can have an architectural form by virtue of either an explicit ArcForm attribute or through architectural markup minimization.

— A client attribute value satisfies a base attribute whose declared value is NOTATION if it is the name of a notation declared in the meta-DTD and allowed as a value of the base attribute, or if the value of the client attribute is the name of a notation derived from a notation declared in the meta-DTD and allowed as a value of the base attribute.

— A client architectural attribute value satisfies a base attribute whose declared value is IDREF if there is an architectural element with that ID as the value of a client architectural attribute whose base attribute is an ID attribute.

A.3.7 Summary of AFDR support options

These options are used in SGML Extended Facilities system declarations to indicate the capabilities of generic architecture engines.

validate	The engine can validate conformance to the AFDR.
construct	The engine can construct architectural instances of client documents that conform to the AFDR.

A.3.8 Conformance

Documents, meta-DTDs, and architecture engines may all conform to this Annex. In addition, this Annex distinguishes validating architecture engines from conforming architecture engines that do not provide complete validation of documents and meta-DTDs against the provisions of this Annex.

A.3.8.1 Conformance of meta-DTDs

A meta-DTD that uses the AFDRMeta extensions shall so indicate by including the following invalid markup declaration as near as practicable to the start of the meta-DTD and before any use of the AFDRMeta extensions:

```
<!AFDR "ISO/IEC 10744:1997">
```

A meta-DTD that does not use the AFDRMeta extensions is a valid meta-DTD if the declarations it contains are valid SGML markup declarations.

A.3.8.2 Conformance of documents and derived meta-DTDs

Documents and meta-DTDs derived from a conforming architecture should so indicate by including the following statement in the architecture support notation declaration:

```
A base architecture used in conformance with the
Architectural Form Definition Requirements of
International Standard ISO/IEC 10744.
```

A.3.8.3 Conforming architecture engine

Conforming architecture engines may be either general or architecture specific. A general architecture engine supports architectural processing independent of the semantics of any architecture. Architecture-specific engines provide architectural processing only for those architectures for which they are designed.

If an architecture engine meets the requirements of this sub-clause it is a conforming architecture engine.

Either form of architecture engine may be a validating architecture engine.

A.3.8.3.1 Conformance of documents

A conforming architecture engine shall require its documents to be conforming SGML documents, and shall not prohibit any markup that ISO 8879 and this Annex would allow in such documents.

NOTE 437 For example, an application markup convention could recommend that only certain minimization functions be used, but could not prohibit the use of other functions if they are allowed by the formal specification.

A.3.8.3.2 Conformance of documentation

A conforming architecture engine's documentation shall meet the requirements of this International Standard (see *11.5 Documentation requirements*).

A.3.8.3.3 Application conventions

A conforming architecture engine shall not enforce application conventions as though they were requirements of this International Standard.

NOTE 438 Warnings of the violation of application conventions can be given, but they must be distinguished from reports of architecture errors.

A.3.8.4 Validating architecture engine

A validating architecture engine shall find and report a reportable architectural error (RAE) if one exists, and shall not report an error when none exists. Such engines can optionally report other errors and may warn of conditions that are potentially, but not necessarily, errors.

A.3.8.4.1 Identification of architecture messages

Reports of architecture errors, including optional reports, shall be identified as architecture messages in such a manner as to distinguish them clearly from all other messages, including warnings of potential architecture errors.

A.3.8.4.2 Content of architecture messages

A report of a architecture error, including an optional report, shall state the nature and location of the error in sufficient detail to permit its correction.

NOTE 439 This requirement is worded to allow implementors maximum flexibility to meet their user and system requirements.

A.3.8.5 Architecture system declaration

An architecture system declaration is an SGML system declaration as defined in ISO 8879, modified as indicated in this subclause.

A new parameter, **AFDR support** (*AFDR*), specifies whether or not the system is a validating architecture engine and whether or not it is capable of producing architectural instances. This parameter follows the SDIF support parameter of an SGML declaration when used for architecture engines.

[34] AFDR support options =
    "AFDR",
    ps+,
    "VALIDATE",

```
 ps+
( "YES" |
  "NO" ),
 ps+,
 "CONSTRUCT",
 ps+
( "YES" |
  "NO" )
```

where:

VALIDATE means that all reportable architectural errors will be found and reported.

CONSTRUCT means that the engine is capable of creating architectural instances from client documents.

A.4 Property Set Definition Requirements (PSDR)

This clause states the requirements for the formal definition of the types of information that notation processors make available to HyTime and DSSSL applications. The clause can be referred to as the “Property Set Definition Requirements” (PSDR).

NOTE 440 Property sets are designed to support the HyTime and DSSSL processing and representation of notation-specific data by providing the information needed by those processes. They are not intended as a general model for making notation-specific data available for arbitrary processing.

A.4.1 Concepts and terminology

A “notation processor” operates on source data that is represented in the notation known to it, recognizing the information described by the source data.

NOTE 441 For example, SGML parsers and HyTime engines are notation processors for their respective notations, ISO 8879 (SGML), and ISO/IEC 10744 (HyTime).

A “property set” defines the types of information that can be returned by a notation processor for use by DSSSL and HyTime applications. The property set for a notation that defines every type of information that a processor for that notation can recognize is known as a “complete property set”.

NOTE 442 The SGML property set and the HyTime property set defined in this International Standard are complete property sets. Whether complete property sets can be defined for other notations depends on how closely the notations resemble SGML and HyTime.

A “grove construction process” uses a notation processor to recognize instances of “classes” and their “properties” as defined in a property set, and represents the recognized instances as “nodes” in a graph structure known as a “grove”.

NOTE 443 Grove is an acronym for “Graph Representation Of property ValuEs”.

NOTE 444 For example, an SGML parser is a process that operates on source data that is represented in the SGML notation defined in ISO 8879. An SGML grove construction process uses an SGML parser to recognize instances of the classes and properties defined in the SGML property set, and then represents the recognized instances as nodes in an SGML grove.

There are two types of grove construction process: “primary” and “auxiliary”. A primary grove construction process operates on data represented in a particular notation and produces a grove known as a “primary grove”. An auxiliary grove construction process operates on nodes in other groves and produces a grove known as a “auxiliary grove”. Property sets used by primary grove construction processes are called “primary property sets”; property sets used by auxiliary grove construction processes are called “auxiliary property sets”.

A grove construction process need not recognize and include in groves all instances of all classes and properties defined in the result grove's property set. A “grove plan” is used to specify which of the classes and properties within a property set are to be included in (or excluded from) a grove.

A grove plan that includes all of the classes and properties in a property set is known as a “complete grove plan”. A grove constructed in accordance with a complete grove plan is known as a “complete grove”.

NOTE 445 The completeness of a grove is in relation to its property set; the completeness of a property set is in relation to its notation.

A.4.1.1 Property sets

A property set is defined in a “property set definition document”, which is an SGML document that conforms to the “property set definition architecture” defined in this clause. The property set definition architecture meta-DTD is identified by the public identifier “ISO/IEC 10744:1997//NOTATION AFDR ARCBASE Property Set Definition Architecture//EN”.

Note that, although they are technically architectural forms, the element types and attribute definitions that appear in this clause are normally used directly as the document type of property set definition documents. They are therefore referred to as element types in this document.

A property set consists of “property set components”: classes, properties, enumerated values (of properties), and normalization rules (for comparing string property values). Some property set components are deemed to occur in all property sets; they are known as “intrinsic components”. The property set definition architecture defines an element type for each component type; elements conforming to those types define components of the corresponding types.

The components of a property set can be organized into one or more separate “modules”. Modules allow classes and properties to be grouped to reflect important distinctions among them that might be a basis for including or excluding them from grove plans.

NOTE 446 In the SGML property set, these distinctions include:

— the applicable specification document (that is SGML, DSSSL, or HyTime)

— whether the property exists only when an optional facility is used (e.g. implicit link); and

— whether the property relates to the source data (e.g. general delimiters), to the abstraction described by the source data (e.g. elements and attributes), or to both (e.g. data characters).

A.4.1.2 Classes and properties

A class identifies a type of information, giving it a name and “providing” an ordered set of properties. Each property is also given a name, as well as a “property number”.

Class names are unique within the property set that defines them; property names are unique within the class that provides them.

Property numbers are assigned to the properties provided by a class according to the order of their definition within the property set definition document. The first property provided by a class is assigned the number 1; the number of each subsequently defined property is determined by adding one to the number of the nearest previously defined property provided by the same class.

NOTE 447 For the purpose of determining the property number, it is immaterial in which module a property definition occurs, or whether it occurs in the content of a class definition or independently.

NOTE 448 Because intrinsic property definitions are considered present at the beginning of each property set definition document, the first few properties of every class are intrinsic properties; this has the effect that the property numbers assigned to the intrinsic properties are the same for all classes.

Property numbers remain constant for all groves, regardless of which properties are included in any given grove plan.

A property has a “declared datatype” which specifies the type of value nodes are permitted to exhibit for the property. A declared datatype is one of the following:

node A node.

nodelist An ordered list of zero or more nodes.

nmndlist An ordered list of zero or more nodes, in which each node exhibits a “name property” whose value uniquely identifies the node within the list. A named node list is also called a “name space”. The datatype of the name property must be string or node, and all of the name properties within a single named node list must have the same datatype.

Note that in effect, there are two varieties of named node list, “string-named” and “node-named”. In the first, each node is identified by a string; in the second, each node is identified by another node. Collecting all the values of the “name” property of the nodes in a string-named node list yields a list of names that are unique within the list. Similarly, collecting all the values of the name properties of the nodes in a node-named node list yields a list of nodes that are unique within the list.

enum A value which represents one of an enumerated set of values; an enumerator.

char An abstract character. The concrete representations of “char” values in groves must allow any “char” value to be distinguished from all other “char” values, and for the semantics of each “char” to be determined. The concrete representation may or may not rely on the DCS, UNICODE, and/or other character sets and repertoires.

string An ordered list of zero or more abstract characters.

strlist An ordered list of zero or more strings.

integer An integer.

intlist An ordered list of zero or more integers.

boolean A boolean value.

compname A property set component name.

cnmlist An ordered list of zero or more component names.

The node, enum, character, string, integer, and compname datatypes are known as “primitive datatypes”; the nodelist, strlist, intlist, and cnmlist datatypes are known as “list datatypes”.

NOTE 449 The string datatype, though described as a list of characters, is still considered a primitive datatype.

A property whose declared datatype is node, nodelist, or nmndlist is said to be “nodal”. Properties with other declared datatypes are “non-nodal”.

A property whose declared datatype is string may have an associated “normalization rule”, which should be consulted when making comparisons involving values exhibited for the property, including looking up a node by its name within a string-named node list.

A property definition can specify conditions under which the property will not apply to an instance of the class that provides it. A node for which a property is inapplicable exhibits a “null value” for the property. (Note that this is not the same as exhibiting an empty value for a list property.)

A.4.1.3 Nodes

A node is an ordered set of “property assignments”, each of which associates a property name with a “value”. The property names are the names of the properties provided by the node's class, the ordering of which determines the ordering of the property assignments.

A node is said to “exhibit” a value “for” a property if that node has a property assignment associating a value with the name of that property. Informally, the “properties of a node” are the properties for which the node exhibits a value. A node is said to be the “owner” of such properties.

A.4.1.4 Groves

Nodes exist in a directed graph in which each node may be connected to other nodes by labeled arcs. For each node occurring in the value exhibited by a node for a nodal property, there is one such arc from the exhibiting node to the value node, the label for which is the name of the nodal property.

NOTE 450 Informally, a node “points at” another node by having the node occur in the value of one of its properties.

The nodes occurring in the value of a nodal property are related to the node that exhibits the property by one of three possible “node relationship types”: “subnode”, “irefnode” (internal-reference node), or “urefnode” (unrestricted-reference node). Nodal properties can be qualified by their associated relationship type as “subnode properties”, “irefnode properties”, or “urefnode properties”; likewise, the arcs in the graph can be qualified by the relationship type assigned to the property to which the arc corresponds.

A node occurring in the value of a subnode property is called a “subordinate node” (or subnode) with respect to the node exhibiting the value, and the node exhibiting the value is the called the “origin” of the subnode. Each node may occur in the value of only one subnode property. The “siblings” of a node occurring in the value of a subnode property are the other nodes, if any, occurring in the value of the same subnode property. The subnode arcs of the graph connect nodes together into “subnode trees”.

Irefnode, or “internal referenced node”, arcs further connect nodes within one subnode tree. Such arcs may cause the graph to contain cycles and convergences.

A grove is a set of nodes connected together as a subnode tree and further connected by the irefnode arcs between the nodes. Each grove has exactly one node that is the root of the subnode tree; this node is the “grove root”, and it is the only node within the grove that has no origin.

The remaining category of nodal properties (and therefore arcs) is urefnode, or “unrestricted referenced node”. A urefnode arc connects a node to nodes in the same or other groves. A set of groves thus connected is a “hypergrove”.

NOTE 451 A hypergrove should not be confused with a hyperdocument or the hypergrove that may result from the processing of a hyperdocument. Any set of groves connected by urefnode arcs is a hypergrove, not just those groves resulting from the processing of hyperdocuments.

A.4.1.5 Content trees

At most one property of a node is designated the “content property” of the node. The content property of a node must be either a subnode property or a string or character property.

If the content property of a node is a subnode property, the property is also the “children property” of the node. The nodes occurring in the value exhibited for the children property of a node are called the “children” of the node, and the node itself is called the “parent” of the children. A node that has children but does not have a parent is a “content tree root”. The set of nodes reachable from a content tree root through children properties forms an ordered tree called a “content tree”.

NOTE 452 If a node has a parent, then the parent is also the node's origin.

NOTE 453 The fact that a grove could contain a collection of disjoint content trees is the reason why it is called a grove.

NOTE 454 Tree addressing of a grove is normally based on content trees rather than on the subnode tree. That is because subnode tree ordering depends on the order of property definitions while content tree ordering depends on the order of node lists in property values -- that is, on the real information represented by the grove.

For this reason, when referring to a grove the unmodified term “tree” means “content tree”.

If the content property of a node is a character or string property, the property is also known as the “data property” of the node. The data of the node is the value of the data property. The data of a node with a children property is the data of each of its children, separated by the value of the node's “data separator property”, if it has one.

A.4.1.6 Grove plan application

For any given processing context, a grove plan may be used to specify what classes of node and what properties are significant within that context. This information can be used to prevent a process from wasting resources on inconsequential data while constructing a grove, or may be used as a mask over an existing grove, hiding those parts that are not of interest.

NOTE 455 The syntax used to specify a grove plan may vary. For instance, HyTime and DSSSL each define ways to specify grove plans (see *7.1.4.1 Grove Plan*).

The effect of applying a grove plan is to remove, from the complete grove theoretically constructible by a given grove construction process with a given grove source, all instances of classes and properties not included by the grove plan.

When an instance of a class (that is, a node) is removed from a grove:

— The node is removed from the value exhibited by any node for any nodal property in which it appears. If the property is a node property (that is not a list property), a null value is then exhibited for the property.

— If the node has a children property, and the reason the node is being removed from the grove is only that its class has been excluded from the grove plan:

- If the class specifies that content trees rooted at nodes of the class are to be “pruned” from the grove, each node in the value of the children property of the node is removed from the grove.
- If the class specifies that content trees rooted at nodes of the class are not to be pruned from the grove, the node or list of nodes exhibited as the value of the node's children property are inserted into the subnode property value in which the node occurred, at the point at which it occurred.

— If the node has a children property, and is being removed for a reason other than that its class has been excluded by the grove plan, each node in the value of the children property is removed from the grove.

— Each node occurring in the value exhibited by the node for non-children subnode properties is removed from the grove.

When an instance of a property (that is a property assignment) is removed from a node:

- The name of the property is removed from the value exhibited by the node for the “all property names” intrinsic property.
- If the property is a subnode property, the name of the property is removed from the value exhibited by the node for the “subnode property names” intrinsic property.
- If the property is the node's children property, the value exhibited by the node for the “child property name” intrinsic property is replaced by a null value.
- If the property is the node's data property, the value exhibited by the node for the “data property name” intrinsic property is replaced by a null value.
- If the property is the node's data separator property, the value exhibited by the node for the “data separator property name” intrinsic property is replaced by a null value.
- If the property is a subnode property, the nodes occurring in the value exhibited for the property are removed from the grove.
- If the property is a non-nodal property, the property is removed from the list of properties of the relevant nodes.

A.4.2 Property set definition architecture

A property set is expressed formally as an SGML element conforming to the element type **property set definition** (*propset*).

The attribute **notation specification document** (*nsd*) is a notation name whose declaration identifies the document that contains the specification of the source data notation. It must be specified on primary property set definitions (those used in constructing primary groves); it may not be specified on auxiliary property set definitions (those used in constructing auxiliary groves).

The attribute **grove construction specification document** (*gcsd*) is a notation name whose declaration identifies the document that contains the specification of the grove construction process. It must be specified if the notation specification document does not include a grove construction process specification, or if the property set definition is for an auxiliary property set.

```
                   <!-- Property Set Definition -->
<!element
   propset        -- Property set definition --
                  -- Clause: A.4.2 --
   - O
   ((desc|note)*,(classdef|normdef)*,psmodule*)

-- Attributes: clausnot, propset --
>
<!attlist
   propset        -- Property set definition --
                  -- Clause: A.4.2 --

   nsd            -- Notation specification document --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: auxiliary property set --
                  -- Constraint: must be specified for primary
                     property sets; may not be specified for
                     auxiliary property sets --

   gcsd           -- Grove construction specification document --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: nsd --
                  -- Constraint: must be specified for auxiliary
                     property sets --
>
```

The attribute form **clause notation** (*clausnot*) specifies the notation used to identify the clause. The clause notation may be specified for the entire property set, for a component, or for a specific property within a component.

The property sets defined in this International Standard use the notation defined in this subclause.

The clause is identified with a character string, as follows:

— The first character is the clause number, the second identifies the sub-clause, the third the sub-sub-clause and the fourth the sub-sub-sub-clause, with 0 representing the absence of any level of the clause structure.

— The final number identifies the paragraph number. Productions, notes and list items are counted as separate paragraphs.

— References to figures are in the form “FIGn”, where “n” is the figure number.

— Any number that exceeds 9 is replaced by a hexadecimal letter.

NOTE 456 There is no provision for numbers greater than 15.

— References to definition clauses of International Standards are in the form 4xxxy where xxx is the decimal sub-clause number and y is the paragraph number.

— References to clauses in annexes of International Standards are prefixed with “X.”, where “X” is the letter of the annex.

NOTE 457 This format is a superset of that used in clause 6.3 of ISO/IEC 13673:1994, which defines how the components of ISO 8879 should be identified within conformance tests. The provisions for definition clauses and annexes do not occur in ISO/IEC 13673.

```
                       <!-- Clause Notation -->
<!attlist
-- clausnot --    -- Clause notation --
                  -- Clause: A.4.2 --
   (propset,classdef,propdef,normdef)

   clausnot       -- Clause notation --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: notation described in A.4.2 --
>
```

A.4.2.1 Shared constructs

A property set is composed of “property set components”: classes, properties, normalization rules (for comparing string property values), and enumerated values (of properties). A property set definition, therefore, consists of property set component definitions. Syntactically, each property set component definition is an element; the type of the element is determined by the type of property set component it defines.

Several attributes are provided by more than one property set component definition element type. These attributes have to do with the naming of components, and with the specification document that describes the components. There are also two descriptive element types that are used in the definitions of multiple types of components.

A.4.2.1.1 Component names

Names of property set components are used in HyTime location address elements and DSSSL expressions. They could also be used for internal communication among programs, in documentation, and for other purposes. Because of this diversity of uses, up to three name attributes are defined for each component:

rcsnm
The attribute **reference concrete syntax (RCS) name** (*rcsnm*) specifies a form of name that complies with the SGML reference concrete syntax.

appnm
The attribute **application name** (*appnm*) specifies a form of name that is less restrictive than an RCS name, but still capable of being bound to a programming language syntax as a name. For example, the application name “processing instruction”, when bound to a programming language, might become “ProcessingInstruction”, “processing-instruction”, or “PROCESSING_INSTRUCTION”, depending on the language.

fullnm
The attribute **full name** (*fullnm*) specifies an unrestricted form of name that is suitable for use in documentation.

For the definition element type of a given property set component, either the rcsnm, the rcsnm and fullnm, or all three forms of name are provided. In the latter two cases, the multiple forms comprise a single name space, meaning that while multiple forms of name for the same component can be identical, none of the names of one component can be the same as any of the names of another component of the same type.

```
                       <!-- Component Names -->
<!attlist
-- rcsnm --       -- Name in RCS --
                  -- Clause: A.4.2.1.1 --
   (enumdef,normdef,propdef,psmodule)

   rcsnm          -- Name in RCS --
      NAME        -- Constraint: unique in component name space --
      #REQUIRED
>
<!attlist
-- appnm --       -- Application name --
                  -- Clause: A.4.2.1.1 --
   (classdef,enumdef,normdef,propdef,psmodule)

   appnm          -- Application name --
      CDATA       -- Constraint: unique in component name space --
      #IMPLIED    -- Default: rcsnm --
>
<!attlist
-- fullnm --      -- Full name --
                  -- Clause: A.4.2.1.1 --
   (classdef,enumdef,normdef,propdef,psmodule)

   fullnm         -- Full name --
      CDATA       -- Constraint: unique in component name space --
      #IMPLIED    -- Default: appnm --
>
```

A.4.2.1.2 Specification and clause

The attribute form **specification and clause** (*spec*) consists of attributes that associate a component with a specification document and clause in that document. The attribute form **clause notation** can be used to specify the clause specification notation if it is different from the overall clause notation specified for a property set.

The attribute **specification document** (*sd*) is a notation name whose declaration identifies the document that contains the specification of the property set component.

The default specification document is the notation specification document for a primary property set, or the grove construction specification document for an auxiliary property set.

The attribute **clause of specification document** (*clause*) identifies the clause of the specification document that defines the component.

```
                  <!-- Specification and Clause -->
<!attlist
-- spec --        -- Specification and clause --
                  -- Clause: A.4.2.1.2 --
   (classdef,propdef,normdef)

   sd             -- Specification document --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: primary property set's notation
                     specification document; auxiliary property set's
                     grove construction specification document. --

   clause         -- Clause of specification document --
      CDATA       -- Constraint: in notation specified by clausnot --
      #IMPLIED    -- Default: if propdef, same clause as classdef;
                     otherwise none --
>
```

A.4.2.1.3 Descriptive elements

Elements of type **description** (*desc*) describe property set components.

Elements of type **explanatory note** (*note*) can be used to supplement a description.

```
                    <!-- Descriptive Elements -->
<!element
   desc           -- Description of property set component --
                  -- Clause: A.4.2.1.3 --
   - O
   (#PCDATA)
>
<!element
   note           -- Explanatory note --
                  -- Clause: A.4.2.1.3 --
   - O
   (#PCDATA)
>
```

A.4.2.1.4 Member of default grove plan

The attribute **member of default grove plan** (*default*) indicates whether or not the component is included in the default grove plan for the property set.

```
                  <!-- Member of default grove plan -->
<!attlist
-- default --     -- Member of default grove plan --
                  -- Clause: A.4.2.1.4 --
   (psmodule,classdef,propdef)

   default        -- Member of default grove plan --
      (default|ndefault)
      ndefault
>
```

A.4.2.2 Modules

Sets of property set components may be grouped together as children of elements of type **property set module** (*psmodule*), so that they may be included or excluded from grove plans as a group. Property set components occurring outside of modules are required in every grove plan; they are automatically included in and may not be excluded from any grove plan.

NOTE 458 Uses of normalization rules, which are not affected by grove plans, are also not affected by whether they are defined within or outside of modules.

The attribute **modules depended on** (*dependon*) identifies modules whose components must be included in any grove plan in which a component of this module is included. Dependencies of depended-on modules are also considered depended upon by a module, though they need not be explicitly named in the value of the module's dependon attribute.

```
                     <!-- Property Set Module -->
<!element
   psmodule       -- Property set module --
                  -- Clause: A.4.2.2 --
   - O
   ((desc|note)*,(classdef|propdef|normdef)*)

-- Attributes: appnm, default, fullnm, psmodule, rcsnm --
>
<!attlist
   psmodule       -- Property set module --
                  -- Clause: A.4.2.2 --

   dependon       -- Modules depended on --
      NAMES       -- Constraint: RCS names of modules defined in this
                     property set --
      #IMPLIED    -- Default: none --
>
```

A.4.2.3 Class definition

An element of type **class definition** (*classdef*) defines a class of node.

NOTE 459 The rcsnm attribute of classdef has the semantics defined for the rcsnm attribute list form. It is redefined here so that the rcs names of class definitions can be SGML unique identifiers, allowing SGML processors to verify that references to class names are valid.

The attribute **content property** (*conprop*) identifies the content property of the class.

The attribute **prune content tree** (*prune*) specifies whether when the class is omitted from a grove plan, the content trees rooted at nodes of the class in a complete grove are pruned away, or whether that branch is merely “grafted” by replacing the node in its parent's children property with the node's children.

The attribute **data separator property** (*dsepprop*) identifies the data separator property of the class.

The attribute **may add** (*mayadd*) is used by the DSSSL transformation language. It is fully described in the DSSSL standard.

NOTE 460 A rule of thumb for its use is that specifying “mayadd” on a class definition indicates that nodes of that class represent information about aspects of the representation of the grove in source form that can be automatically managed by the process that converts the grove into source form.

```
                      <!-- Class Definition -->
<!element
   classdef       -- Class definition --
                  -- Clause: A.4.2.3 --
   - O
   ((desc|note)*, propdef*)

-- Attributes: appnm, clausnot, default, fullnm, spec --
>
<!attlist
   classdef       -- Class definition --
                  -- Clause: A.4.2.3 --

   rcsnm          -- Name in RCS --
      ID          -- Constraint: unique in component name space --
      #REQUIRED

   conprop        -- Content property --
      NAME        -- Constraint: RCS name of nodal property of class
                     with noderel=subnode, char property, or string --
      #IMPLIED    -- Default: no content property --

   prune          -- Prune content tree --
      (prune|nprune)
                  -- Constraint: can only be nprune if the set of
                     allowed classes of the content property is
                     guaranteed to be a subset of that of parent's
                     content property --
      prune

   dsepprop       -- Data separator property --
      NAME        -- Constraint: RCS name of char or string property
                     of class --
      #IMPLIED    -- Default: none --
   mayadd         -- May add --
      (mayadd|mayntadd)
      mayntadd
>
```

A.4.2.4 Property definition

An element of type **property definition** (*propdef*) defines a property.

The attribute **class name** (*cn*) identifies the provider of the property. Its value must be the RCS name of a class defined within the same module, within one of that module's depended on modules, or outside the scope of any module. If the property definition occurs within the content of a class definition, the specification of class name may be omitted; such omission implies that the property is provided by the class within whose definition the property's definition occurs.

An exception is made for the definition of intrinsic properties (see *A.4.3 Intrinsic properties*); there the class name may be specified as “#ALL”, where “#ALL” means that the property is provided to nodes of all classes. Use of this keyword in this context may occur only within this International Standard; it may not occur in any actual property set definition.

The attribute **property value datatype** (*datatype*) specifies the datatype of values exhibited for the property.

If the datatype is nodal, the attribute **node relationship** (*noderel*) indicates the relationship between the property owner and the nodes in the property value. Also, the attribute **allowed classes** (*ac*) specifies the classes of nodes allowed in the property value. Unless the node relationship is urefnode, the complete set of allowed classes must be specified. Allowed classes may be specified for a urefnode property only if it is intended to refer only to nodes of the same property set (though those nodes may be in other groves). Urefnode property definitions that do not specify allowed classes define properties whose value nodes may be of any class from any property set, and may be in any grove.

When the datatype is nmndlist the attribute **allowed classes name property** (*acnmprop*) must specify for each allowed class the name of the property that is to serve as the name property for nodes of that class with respect to the named node list.

If the primitive type of the datatype is “string”, the attribute **string normalization rule** (*strnorm*) identifies the normalization rule applicable to the property.

The attribute **verify type** (*vrfytype*) is used by the DSSSL transformation language. It is fully described in the DSSSL standard.

NOTE 461 The following are rules of thumb for its use:

— A verify type of “optional” specified on a property definition indicates that values exhibited for the property represent information about aspects of the representation of the grove in source form that can be automatically managed by the process that converts the grove into source form.

— A verify type of “derived” specified on a property definition indicates that values exhibited for the property are derivable from other property values within the grove.

If a property definition element contains an element of the type **when property exists** (*when*), any node that does not meet the conditions specified by the content of the element will exhibit a null value for the property.

```
                     <!-- Property Definition -->
<!element
   propdef        -- Property definition --
                  -- Clause: A.4.2.4 --
   - O
   ((desc|note)*,when?,enumdef*)

-- Attributes: appnm, clausnot, default, fullnm, rcsnm, spec --
>
<!attlist
   propdef         -- Property definition --
                   -- Clause: A.4.2.4 --

   cn              -- Class name --
                   -- Provider of property --
      IDREF        -- Constraint: RCS name of class in property set --
      #IMPLIED     -- Default: containing classdef --
                   -- Constraint: must be specified if propdef occurs
                      outside of a classdef --

   datatype        -- Datatype of property value --
      (node|nodelist|nmndlist|enum|char|string|strlist|integer|
       intlist|boolean|compname|cnmlist)
      #REQUIRED

   noderel         -- Node relationship --
      (subnode|irefnode|urefnode)
      #IMPLIED     -- Default: not nodal --
                   -- Constraint: required if datatype is nodal --

   ac              -- Allowed classes --
      IDREFS       -- Constraint: RCS names of classes defined in
                      property set --
      #IMPLIED     -- Default: not nodal or if noderel=urefnode, any
                      class in any property set --
                   -- Constraint: required if datatype is nodal, unless
                      noderel=urefnode --

   acnmprop        -- Allowed class name properties --
      NAMES        -- Constraint: one property RCS name for each
                      allowed class; must be either all string
                      properties or all node properties of their
                      respective classes --
      #IMPLIED     -- Default: not nmndlist --
                   -- Constraint: required if datatype is nmndlist,
             otherwise, may not be specified. --

   strnorm         -- String normalization rule --
       NAME        -- Constraint: RCS name of normdef defined in
                      property set --
       #IMPLIED    -- Default: none --
                   -- Constraint: may be specified only if datatype is
                      string or strlist --

   vrfytype        -- Verify type --
       (derived|optional|other)
       other
>
<!element
   when            -- When property exists --
```

```
                  -- Clause: A.4.2.4 --
                  -- Condition precedent for a non-null value to be
                     exhibited --
   - O
   (#PCDATA)
>
```

A.4.2.4.1 Enumerated value definition

An element of the type **enumerated value definition** (*enumdef*) defines an allowed value of the property in the content of whose definition it appears. The datatype of the property must be enum.

```
                 <!-- Enumerated value definition -->
<!element
   enumdef        -- Enumerated value definition --
                  -- Clause: A.4.2.4.1 --
   - O
   (desc|note)*

-- Attributes: appnm, fullnm, rcsnm --
>
```

A.4.2.5 Normalization rule definition

The element type **normalization rule definition** (*normdef*) defines a string normalization rule by referring to it in a specification document.

```
                <!-- Normalization Rule Definition -->
<!element
   normdef        -- Normalization rule definition --
                  -- Clause: A.4.2.5 --
   - O
   (desc|note)*

-- Attributes: appnm, clausnot, fullnm, rcsnm, spec --
>
```

A.4.3 Intrinsic properties

The properties defined in this clause, known as the “intrinsic properties”, are included in every property set. The intrinsic property definitions may not be explicitly stated within any property set definition; however, they are deemed to occur at the start of every property set definition, in the order they occur in this clause.

Note that, because intrinsic properties are silently included in all property sets, their names occur in the property name spaces of all classes. This means that the names of non-intrinsic properties may not duplicate those of intrinsic components.

The following intrinsic properties are exhibited by all nodes in all groves:

```
<propdef cn="#ALL" rcsnm=classnm appnm="class name" compname>
<desc>
The name of the node's class.

<propdef cn="#ALL" rcsnm=grovroot appnm="grove root" node irefnode>
```

```
<desc>
The root of the grove in which the node occurs.
<note>
The value exhibited by a node for this property shall be a reference
to the node itself if the node is the root of a grove.

<propdef cn="#ALL" rcsnm=subpns appnm="subnode property names"
cnmlist>
<desc>
The names of all the subnode properties for which values are exhibited
by the node, in ascending order of property number.
<note>
The names of properties excluded from the grove plan are also excluded
from the value of this property.

<propdef cn="#ALL" rcsnm=allpns appnm="all property names" cnmlist>
<desc>
The names of all the properties for which values are exhibited by the
node, in ascending order of property number.
<note>
The names of properties excluded from the grove plan are also excluded
from the value of this property.

<propdef cn="#ALL" rcsnm=childpn appnm="children property name"
compname>
<desc>
The name of the children property of the node.
<when>
The node has a children property that is included by the grove plan.

<propdef cn="#ALL" rcsnm=datapn appnm="data property name" compname>
<desc>
The name of the data property of the node.
<when>
The node has a data property that is included by the grove plan.

<propdef cn="#ALL" rcsnm=dseppn appnm="data sep property name"
fullnm="data separator property name" compname>
<desc>
The name of the data separator property of the node.
<when>
The node has a data separator property that is included by the grove
plan.

<propdef cn="#ALL" rcsnm=parent node irefnode>
<desc>
The parent of the node.
<when>
The node has a parent.

<propdef cn="#ALL" rcsnm=treeroot appnm="tree root"
fullnm="content tree root" node irefnode>
<desc>
The root of the content tree in which the node occurs.
<note>
The value exhibited by a node for this property shall be a
```

```
reference to the node itself if the node is the root of a content
tree.

<propdef cn="#ALL" rcsnm=origin node irefnode>
<desc>
The origin of the node.
<when>
The node has an origin.

<propdef cn="#ALL" rcsnm=otsrelpn
appnm="origin to subnode rel property name"
fullnm="origin to subnode relationship property name"
compname>
<desc>
The name of the subnode property of the node's origin, the value
exhibited for which includes the node.
<when>
The node has an origin.
```

The principal tree root property is exhibited by every node of every grove, though the value exhibited for it may be null if the grove has no principal tree root. Unlike the properties defined above, the value of the principal tree root property is dependent on the grove construction process; if a grove is to have a principal tree, the grove construction specification document must specify which of the grove's content trees it is.

NOTE 462 A grove may include one or more content trees and still not have a principal tree; the presence of a principal tree must be stated explicitly by the grove construction specification document.

NOTE 463 In an SGML grove, the principal tree root is the document element.

```
<propdef cn="#ALL" rcsnm=ptreert appnm="principal tree root"
node irefnode>
<desc>
The root node of the principal content tree.
<when>
The grove has a principal content tree.
```

The source property applies only to auxiliary groves; it is not exhibited by nodes in primary groves, and its definition is not deemed to be included in primary property set definitions.

NOTE 464 This means that in primary property sets, “source” is not a reserved property name, nor is a property number allocated for it.

```
<propdef cn="#ALL" rcsnm=source nodelist urefnode>
<desc>
The nodes in the source grove from which the node was derived.
<note>
This property is exhibited only by nodes occurring in auxiliary groves.
```

A.4.4 Useful grove construction processes

This clause defines grove construction processes of general utility to applications.

A.4.4.1 Value-To-Node (VTN) grove construction

One of the intended uses for groves is as an abstraction for locating information. Each node in a grove represents a piece of information; the property set according to which the grove was constructed defines the types of information that can be located.

Each node can be further broken down into a list of properties and their values. List property values can be broken down into their constituent members. These also represent pieces of information that may need to be located. Nodal property values present no problem; as the values are nodes, they are already valid locations. However, non-nodal property values are not independent of the nodes that exhibit them, and therefore present the problem that they cannot be located directly.

The “Value-To-Node grove construction process” solves this problem by defining a way to construct groves derived from property values. Nodes thus constructed are considered to be the locations of their source property values.

A.4.4.1.1 The Value-To-Node property set

Each class in the VTN property set corresponds to a primitive non-nodal datatype. Each class has a “source property” property of datatype compname, which in combination with the intrinsic “source” property is used to identify the property value a particular VTN grove represents.

Each class also has a “value” property. If the name of the class is that of a simple datatype, then the datatype of the value property of the class is that simple datatype, with the exception of the “enum” class, whose value property is of datatype compname. If the name of the class is that of a list datatype, then the datatype of the value property of the class is node list (with node relationship “subnode”), and the allowed class for the property is the class corresponding to the simple datatype of which the list datatype is composed. The value properties of the char, string, and list classes are also the content properties of those classes.

The VTN property set is:

```
<!-- Value-To-Node (VTN) Property Set -->

<!DOCTYPE propset
   PUBLIC "ISO/IEC 10744:1997//DTD Property Set//EN"
[
<!NOTATION VTN
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Value-To-Node Grove Construction Process//EN"
>
]>

<propset gcsd="VTN">

<classdef rcsnm=enum fullnm="enumeration">

<propdef rcsnm=srcprop appnm="source property" compname>

<propdef rcsnm=value compname>
<desc>
The name of the enumdef corresponding to the value of the source
property.

<classdef rcsnm=char fullnm="character" conprop=value>

<propdef rcsnm=srcprop appnm="source property" compname>
```

```
<propdef rcsnm=value char>

<classdef rcsnm=string conprop=value>

<propdef rcsnm=srcprop appnm="source property" compname>
<when>
The string is not a member of a string list.

<propdef rcsnm=value string>

<classdef rcsnm=strlist fullnm="string list" conprop=value>

<propdef rcsnm=srcprop appnm="source property" compname>

<propdef rcsnm=value nodelist subnode ac="string">

<classdef rcsnm=integer>

<propdef rcsnm=srcprop appnm="source property" compname>
<when>
The integer is not a member of an integer list.

<propdef rcsnm=value integer>

<classdef rcsnm=intlist fullnm="integer list" conprop=value>

<propdef rcsnm=srcprop appnm="source property" compname>

<propdef rcsnm=value nodelist subnode ac="integer">

<classdef rcsnm=boolean>

<propdef rcsnm=srcprop appnm="source property" compname>

<propdef rcsnm=value boolean>

<classdef rcsnm=compname fullnm="component name">

<propdef rcsnm=srcprop appnm="source property" compname>
<when>
The component name is not a member of a component name list.

<propdef rcsnm=value compname>

<classdef rcsnm=cnmlist fullnm="component name list" conprop=value>

<propdef rcsnm=srcprop appnm="source property" compname>

<propdef rcsnm=value nodelist subnode ac="compname">
```

A.4.4.1.2 VTN groves

VTN groves are constructed from the values of non-nodal properties. The class of the grove root is determined by the datatype of the “source property” property (not to be confused with the intrinsic “source” property); that is, if the datatype of the source property is “integer”, then the class of the grove root is “integer”.

The VTN grove constructed from the value of a property with a simple datatype consists of a single node. The value of the VTN node's intrinsic “source” property is the node that exhibits the source value. The value of the VTN node's “source property” property is the component name of the property for which the source node exhibited the source value. The value of the VTN node's “value” property is the same as the value exhibited by the source node for the source property, except when the source property's datatype is an enumeration, in which case the value of the VTN node's “value” property is the component name of the enumerated value.

Multiple VTN nodes are constructed from the value of a property with a list datatype. They consist of a grove root and a list of its children nodes. The grove root's intrinsic “source” and (not intrinsic) “source property” properties have values as described for simple datatypes above. The value of the grove root's “value” property is a list of children nodes, whose class names are the same as the name of the simple datatype of which the list datatype is composed. Each child node corresponds to one member of the list property value, and the children nodes appear in the same order as the members to which they correspond. The value of each child node's “value” property is identical to the source node's corresponding list property value member.

A.4.4.2 Data tokenizer (DATATOK) grove construction

A datatok grove construction process applies a lexical model with subordinate “match token models” to data in a grove to produce a datatok grove. The datatok process is defined by the following steps:

1) A grove root (a node of class “tokroot”) is created. The value of its “source” property is set to the list of nodes that is the grove source.

2) A list of character strings is created by extracting the data from each of the source nodes.

3) If a “source concatenation separator” is specified, the strings are concatenated together, separated by the specified separator, resulting in a list containing a single string.

4) The lexical model is applied to each string in the list. For each string, a list of the sub-strings that satisfy a match token model within the lexical model is created. The result is a list of sub-lists of these strings. Each sub-string retains any lexicographic reordering performed during the application of the lexical model.

5) If a “token concatenation separator” is specified, the strings of each sub-list are concatenated together, separated by the specified separator, resulting in a list of sub-lists, each containing a single string.

6) Each sub-list of strings is replaced by its contained strings, resulting in a single list of strings.

7) If a “result concatenation separator” is specified, the strings are concatenated together, separated by the specified separator, resulting in a list containing a single string.

8) A list of “tokenstr” nodes is created where each node corresponds to one of the strings in the list and the nodes occur in the order of their corresponding strings. The value of the “string” property of each node is set to the node's corresponding string, and the value of the “source” property of each node is set to the list of nodes in the grove source containing all or part of the node's corresponding string.

9) The value of the “strings” property of the grove root is set to the list of tokenstr nodes.

The following example shows the result of applying a “word” tokenizer with different concatenation options to the data contained by the three sibling nodes resulting from this document fragment (when using the default HyTime grove plan):

```
<state>Oregon</state> <animal>river otters</animal> are cute
```

The list of strings resulting from step 2 is: (“Oregon” “ ” “river otters” “ are cute”). (Note that because this example is mixed content, the second string in the list is the space between the State end-tag and the Animal start tag.)

The results when performing concatenation at different points in the process are shown below (parentheses indicate lists, square brackets indicate result tokenstr nodes):

No concatenation:

Step 4 result: ((“Oregon”) () (“river” “otters”) (“are” “cute”))

Step 6 result: (“Oregon” “river” “otters” “are” “cute”)

Step 8 Tokenstr nodes: [“Oregon”] [“river”] [“otters”] [“are”] [“cute”]

Source concatenation, separation character=“X”**:**

Step 3 result: (“OregonX Xriver ottersX are cute”)

Step 4 result: (“OregonX” “Xriver” “ottersX” “are” “cute”)

Step 6 result: (“OregonX” “Xriver” “ottersX” “are” “cute”)

Step 8 Tokenstr nodes: [“OregonX”] [“Xriver”] [“ottersX”] [“are”] [“cute”]

Token concatenation, separation character=“X”**:**

Step 4 result: ((“Oregon”) () (“river” “otters”) (“are” “cute”))

Step 5 result: ((“Oregon”) () (“riverXotters”) (“areXcute”))

Step 6 result: (“Oregon” “riverXotters” “areXcute”)

Step 8 Tokenstr nodes: [“Oregon”] [“riverXotters”] [“areXcute”]

Result concatenation, separation character=“X”**:**

Step 4 result: ((“Oregon”) () (“river” “otters”) (“are” “cute”))

Step 6 result: (“Oregon” “river” “otters” “are” “cute”)

Step 7 result: (“OregonXriverXottersXareXcute”)

Step 8 Tokenstr nodes: [“OregonXriverXottersXareXcute”]

A.4.4.2.1 Data tokenizer property set

The datatok property set is:

```
<!-- Data Tokenizer Property Set -->

<!DOCTYPE propset
   PUBLIC "ISO/IEC 10744:1997//DTD Property Set//EN"
[
<!NOTATION datatok
   PUBLIC "ISO/IEC 10744:1997//NOTATION
```

```
           Data Tokenizer Grove Construction Process//EN"
>
]>

<propset gcsd=datatok>

<classdef rcsnm=tokroot appnm="tokenized root" conprop=strings>
<desc>
The grove root resulting from a data tokenizer process.

<propdef rcsnm=strings nodelist subnode ac=tokenstr>
<desc>
A node list of token strings

<classdef rcsnm=tokenstr appnm="tokenized string" conprop=string>
<desc>
A string of one or more tokens.

<propdef rcsnm=string string>
```

A.4.4.2.2 Data tokenizer notation form

Data tokenizer grove construction processes are declared as data content notations derived from the data tokenizer (datatok) notation form.

The lexical model used by a data tokenizer process may be inherent to the process (for example, a process that performs lexical analysis of a natural language), or may be specified separately as data in a notation whose definition is included in the definition of the data tokenizer process itself. The lexical model is specified as the content of elements conforming to data tokenizer notations.

NOTE 465 The NotNames attribute of the data attributes for elements facility of the General Architecture may be used to map notation content to an element attribute (see *A.5.3 Data Attributes for Elements (DAFE)*).

The attribute **concatenate source** (*catsrc*) specifies that source concatenation be performed using the specified separator string.

The attribute **concatenate tokens** (*cattoken*) specifies that token concatenation be performed using the specified separator string.

The attribute **concatenate result** (*catres*) specifies that result concatenation be performed using the specified separator string.

The attribute **hit boundary constraint** (*boundary*) specifies the boundary constraints, within the match domain, of a hit that satisfies the model, as follows:

sodeod	The hit must extend from the start of domain to the end.
sodiec	The hit must begin at the start of domain but need not extend to the end of domain.
isceod	The hit must end at the end of domain but need not begin at the start of domain.
isciec	The hit can occur anywhere in the match domain.

inmodel Boundary requirements are expressed within the model.

NOTE 466 The boundary constraints correspond to the “^” (caret) and “$” (dollar) operators in POSIX regular expressions. E.g., “sodeod” corresponds to a regular expression that starts with caret and ends with dollar.

The attribute **maximum token size** (*maxtoksz*) specifies the maximum number of characters that can satisfy a single match token group. It is an error if a match token group for which matching is attempted cannot be matched within the maxtoksz limit.

```
          <!-- Data Tokenizer Grove Construction Process -->
<![ %datatok; [
<!notation
   datatok        -- Data tokenizer grove construction process --
                  -- Clause: A.4.4.2.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Data Tokenizer Grove Construction Process//EN"

-- Attributes [locs]: datatok --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   datatok        -- Data tokenizer grove construction process --
                  -- Clause: A.4.4.2.2 --

   catsrc         -- Source concatenation separator --
      CDATA
      #IMPLIED    -- Default: no source concatenation --

   cattoken       -- Token concatentation separator --
      CDATA
      #IMPLIED    -- Default: no token concatenation --

   catres         -- Result concatenation separator --
      CDATA
      #IMPLIED    -- Default: no result concatenation --

   boundary       -- Hit boundary constraint --
      (sodeod|sodiec|isceod|isciec|inmodel)
      isciec

   maxtoksz       -- Maximum token size --
      NUMBER
      #IMPLIED    -- Default: system defined --
>
]]><!-- datatok -->
```

A.4.4.3 Plain text (PLAINTXT) grove construction

The plain text grove construction process builds groves from data about which all that is known is that it is composed of characters.

A plain text grove consists of a grove root of class **plain text document** (*document*), the value of the **text** property exhibited by which is a list of nodes of class **data character** (*datachar*). One datachar node is created for each character recognized in the source data, setting the value of each datachar node's **character** (*char*) property to the corresponding recognized character. The order of the datachar nodes is determined by the order of their corresponding characters in the source data.

A.4.4.3.1 Plain text property set

The plaintxt property set is:

```
<!-- Plain Text Property Set -->

<!DOCTYPE propset
   PUBLIC "ISO/IEC 10744:1997//DTD Property Set//EN"
[
<!NOTATION plaintxt
   PUBLIC "ISO/IEC 10744:1997//NOTATION Plain Text//EN"
>
]>

<propset nsd=plaintxt>

<classdef rcsnm=document fullnm="plain text document" conprop=text>

<propdef rcsnm=text nodelist subnode ac="datachar">

<classdef rcsnm=datachar appnm="data char" fullnm="data character"
conprop=char>

<propdef rcsnm=char fullnm="character" char>
```

A.4.5 Canonical Grove Representation (CGR)

Groves are an abstract data representation. Processing systems need not reflect a literal grove in their internal data structures. However, a literal representation of groves is useful for validation, debugging, and, potentially, interchange among different applications. The canonical grove representation (CGR) document type is used to create character representations of groves such that there can be exactly one canonical grove representation for a given grove or hypergrove.

The CGR document type is defined as an SGML document type derived from the HyTime architecture augmented by a set of constraints on how source documents are formed. These constraints ensure that the only differences between CGR documents are differences resulting from differences in node property values. When the constraints are adhered to, CGR documents can be meaningfully compared as character strings.

NOTE 467 The constraints on CGR document sources have also been designed to enable parsing of CGR documents using the standard string parsing features of common programming languages. While CGR documents are SGML documents and can be parsed as such, they can also be parsed as a sequence of delimited records.

A.4.5.1 Canonical grove representation document type

The CGR document type consists of the following element types:

grove — Represents a single grove. Contains a single *node* element followed by zero or more *urefloc* elements.

node Represents a single node. Contains as subelements the properties of a node (*nodeprop*). Each node has an ID value assigned to it according to the algorithm defined under *A.4.5.3 Algorithm for assigning IDs to nodes*. The *class* attribute of the node element names the RCS class name for the node.

nodeprop Represents a single property value. The content of the *nodeprop* element is the property value. Node lists are represented by a blank-delimited list of the IDs of the node elements representing the nodes in the node list. Primitive (non-nodal) data types are represented by strings according to the rules in section *A.4.5.2 Constraints on CGR source construction*.

The nodeprop element type has the following attributes:

datatype The RCS name of the property's datatype .

rcsnm The RCS name of the property.

noderel The node relationship type for the property. One of subnode, irefnode, urefnode, or primitive.

isnull Indicates that the property value is null when specified.

urefloc Represents the address of a node in another grove. The *urefloc* element type is derived from the property location address element form (*proploc*). When the CGR document is generated independently of the hypergrove in which it occurs, the content of each *urefloc* element is the string “UNKNOWN”, indicating that the name of the node addressed is not known. When the CGR document is generated as part of the process of generating all the CGR documents for an entire hypergrove, the content of the *urefloc* is the ID of the node in the CGR document that contains the referenced node.

slsep Separates strings in string lists.

CGR documents conform to the reference concrete syntax except that they use a name length of 32.

```
<!--=================================================================
    Canonical Grove Representation Document Type

    ISO/IEC 10744:1997//DTD Canonical Grove Representation//EN

    =================================================================-->

<?IS10744 ArcBase HyTime>
<!NOTATION HyTime
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!ATTLIST #NOTATION HyTime
   ArcFormA NAME     #FIXED HyTime
   ArcNamrA NAME     #FIXED HyNames
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    #FIXED "HyTime"
   ArcBridF NAME     #FIXED HyBrid
   ArcAuto  NAME     #FIXED ArcAuto
   ArcOptSA CDATA    #FIXED "locs"
   locs     CDATA    #FIXED "nmsploc"
>
<!NOTATION AFDRMeta
```

```
   PUBLIC "ISO/IEC 10744//NOTATION AFDR Meta-DTD Notation//EN"
>
<!ENTITY HyTime
   PUBLIC "ISO/IEC 10744//DTD AFDR Meta-DTD
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
   CDATA AFDRMeta
>

<!ELEMENT
   grove          -- Grove document element --

   (node,urefloc*)
>

<!ELEMENT
   urefloc        -- Unrestricted reference node location address --

   (#PCDATA)      -- Constraint: content of urefloc is "UNKNOWN"
                     when CGR document not generated as part of
                     hypergrove CGR generation process. --
>
<!ATTLIST urefloc
   id       ID       #REQUIRED
   HyTime   NAME     #FIXED nmsploc
   namespc  NAME     #FIXED elements
   locsrc   ENTITY   #REQUIRED
   notname  NAME     #FIXED ignore
>

<!ELEMENT
   node           -- Node in the grove --

   (nodeprop*)
>
<!ATTLIST node
   id             -- Unique ID of node --
      ID          #REQUIRED

   class          -- RCSNM name of the class for this node --
      NAME
      #REQUIRED
>

<!ELEMENT
   nodeprop       -- Node property --

   (#PCDATA|slsep|node)*
                  -- Constraint: Nodes allowed in content only when
                     noderel is "subnode". --
>
<!ATTLIST nodeprop
   rcsnm          -- RCS name of property --
      NAME
      #REQUIRED

   datatype       -- The RCSNM of the datatype --
```

```
      NAME     #REQUIRED

   noderel        -- Relationship of nodes in property to node
                     exhibiting property --
      (subnode|urefnode|irefnode|primitive)
      #REQUIRED

   isnull         -- Property is null --
      (isnull)
      #IMPLIED    -- Default: Property is not null --
>

<!ELEMENT
   slsep          -- String list separator --

   - O
   EMPTY
>

<!-- Entities for escaping markup delimiters in data content -->
<!ENTITY lt   "<" -- Left angle bracket -->
<!ENTITY gt   "<" -- Right angle bracket -->
<!ENTITY apos "'" -- Apostrophe (LITA) -->
<!ENTITY amp  "&" -- Ampersand -->
```

A.4.5.2 Constraints on CGR source construction

CGR documents must conform to the following constraints on their source:

— No white space is allowed except as expressly defined in these rules or required by SGML.

— Exactly one SPACE character is used to separate parameters that require separation.

— All keywords are specified in upper case (only element content and CDATA attribute values may be in mixed case).

— The DOCTYPE declaration up to, but not including, the declaration subset open (if any) is specified on a single line as follows:

```
<!DOCTYPE GROVE PUBLIC "ISO/IEC 10744:1997//DTD Canonical Grove Representation//
EN"
```

The markup declaration close is specified on a line by itself.

— If there is a declaration subset (containing entity declarations for other CGR documents), the declaration subset open and declaration subset close delimiters are each specified on lines by themselves. Each entity declaration is specified on a line by itself (see *A.4.5.3 Algorithm for assigning IDs to nodes*).

— Start tags are specified on multiple lines as follows:

- The STAGO and GI are specified on a single line.
- Each attribute is specified on a single line by itself. Attributes are specified in the order they occur in the CGR document type attribute list declarations. Fixed attributes are never specified.
- The TAGC is specified on a line by itself.

— For end-tags, the ETAGO and GI are specified on a single line. The TAGC is specified on a line by itself.

— Each grove document must consist of exactly one entity (the document entity), except for the external declaration subset, which is always referred to by public identifier as described above.

Primitive property values are represented as character data content of nodeprop elements. The primitive data types are represented by strings as follows:

enum — The RCS name of the enumeration value, in upper case.

string — Character data content exactly as in the property value, with the exception that the characters “<” (left angle bracket), “>” (right angle bracket), “’” (appostrophe), and “&” (ampersand) are replaced by references to the general entities “lt”, “gt”, “apos”, and “amp”, respectively. These entities are declared in the CGR docment type.

string list — A list of string values, each value except the last followed by an slsep start-tag.

integer — The string representation of the integer value, base 10.

component name — The RCS name of the component, in upper case.

integer list — Blank-delimited list of integers.

component name list — Blank-delimited lists of component names.

Properties that have no value (because the when clause for the property was not satisfied) exhibit the value “ISNULL” for the isnull attribute.

The IDs for nodes are generated according to the algorithm defined in the next subclause.

A.4.5.3 Algorithm for assigning IDs to nodes

The iref and uref relationship types are represented as SGML ID references in CGR documents. The IDs for nodes must be generated as follows:

— Initialize a counter to zero.

— For each node element, generate its ID by prepending the string “X” to the string value (base 10) of the counter. Increment the counter by one.

— For each urefnode arc to a node, generate that node's ID as for internal nodes. In addition, generate a corresponding urefloc element with that ID. Urefloc elements are listed following the root node in the order they were generated (thus, in the order of first reference to the nodes they represent).

When a CGR document is generated in isolation, it is impossible to acquire the IDs of nodes in other CGR documents for use in the content of urefloc elements. It is also impossible to generate entity names for the CGR documents that would contain those nodes. However, when a CGR document is generated as part of the generation of the canonical grove representation for a complete hypergrove, then the content of urefloc elements must be the IDs assigned to those nodes. In addition, an entity must be declared for each different grove to which unrestricted references are made.

These entities are declared as follows:

— Initialize a counter to zero

— For each different grove to which an unrestricted reference is made, in the order of first reference, generate an entity name by prepending the string “CGR” to the string value (base 10) of the counter. Increment the counter by one.

— For each entity name generated, in the order of generation, create an entity declaration with the following form:

```
   <!ENTITY name PUBLIC "-//HYPERCGR//DOCUMENT name//EN" CDATA HyTime
   >
```

— Where both occurrences of “name” are replaced by the generated entity name.

NOTE 468 There is no requirement, and no effort should be made, to coordinate the public identifiers generated in different grove documents. Thus the same grove document may be referred to by different public identifiers in the different groves that refer to it.

Each entity declaration is included in the internal DOCTYPE declaration subset on a single line, with the declaration close on a line by itself, as shown above.

NOTE 469 It is up to the system that generates a canonical hypergrove to create the mapping catalog that associates the public identifiers generated for grove documents with those documents.

A.4.6 Conformance

A document or system can support the use of property sets and groves in accordance with the requirements of this annex without conforming to other requirements of this International Standard.

A system that conforms to these property set definition requirements shall so indicate by including the following statement in a property set definition document that shall be maintained as part of the system's conforming documentation:

```
An implementation conforming to the
Property Set Definition Requirements
of International Standard ISO/IEC 10744.
```

A.5 General Architecture

The General Architecture provides a number of element, attribute, and notation forms of potential use to any SGML architecture. A General Architecture engine (as opposed to a generic architecture engine) creates a semantic grove that reflects the original document completely and adds to it nodes and properties specific to the General Architecture facilities. This grove uses the SGML property set, which includes the class and property definitions required by the General Architecture.

The General Architecture may be used as a base architecture for client documents or for derived architectures.

NOTE 470 For example, HyTime is derived from the General Architecture and includes all the General Architecture facilities.

A.5.1 General Architecture Declaration Template

The general form of the General Architecture support declarations are as follows:

```
          <!-- General Architecture Support Declarations -->
     <!-- TEMPLATE FOR DECLARATION IN DTD or DERIVED META-DTD -->
<!Notation
   GenArc         -- General Architecture --
                  -- A base architecture used in conformance with the
                     Architectural Form Definition Requirements of
                     International Standard ISO/IEC 10744. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           General Architecture//EN"
>
<!Attlist #NOTATION
```

```
   GenArc         -- General Architecture --

   ArcFormA NAME     GAForm
   ArcNamrA NAME     GANames
   ArcSuprA NAME     GASupr
   ArcIgnDA NAME     GAIgnD
   ArcDocF  NAME     GADoc
   ArcDTD   CDATA    "GenArc"

   -- NAMELEN must be at least 9 because the General Architecture
      meta-DTD uses 8-character parameter entity names. --
   ArcQuant CDATA    #FIXED "NAMELEN 9"

   ArcDataF NAME     GABridN
   ArcBridF NAME     GABrid
   ArcAuto  (ArcAuto|nArcAuto) ArcAuto
   ArcOptSA NAMES    #FIXED "commatts dcnatts"

   commatts       -- Common element attribute options --
      CDATA       -- Lextype: csname+ --
                  -- Constraint: lists parameter entities in meta-DTD
                     to be set to "INCLUDE" --
      "dafe dvlist ireftype HyLex HyOrd lextype opacity REGEX"

   dcnatts        -- Common data attribute options --
      CDATA       -- Lextype: csname+ --
                  -- Constraint: lists parameter entities in meta-DTD
                     to be set to "INCLUDE" --
      "altreps included superdcn"

   irefmodl       -- SGML model groups for ireftype --
                  -- Clause: A.5.5 --
      (SGMLmdl|nSGMLmdl)
      nSGMLmdl


>
<!NOTATION AFDRMeta
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD Notation//EN"
>
<!Entity
   GenArc         -- General Architecture meta-DTD --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD
           General Architecture//EN"

   CDATA AFDRMeta
>
```

A.5.2 Common attributes of elements

The attribute forms *id*, *etfullnm*, and *opacity* consist of common attributes that respectively identify an element, its full name, and whether or not it is to be considered contextually “invisible” to its parent element.

The attribute **element unique identifier** (*id*) specifies a unique name for the element within the document in which it occurs.

NOTE 471 Thus, when the General Architecture is in effect, all attributes with a value prescription of “ID” must either be named “id” or must have their names remapped to “id” using the General Architecture renaming attribute (GANames). This requirement is normally met automatically by the rules by which applications are related to architectures (see *A.3.6.4 Relating applications and architectures*).

The attribute **element type full name** (*etfullnm*) specifies an unrestricted form of name that is suitable for use in documentation.

NOTE 472 The attribute makes the name available to programs.

The attribute **opacity** (*opacity*) indicates whether the element is opaque or transparent with respect to its content. A transparent element is one whose content (if any) must be valid in the context in which the element occurs, in both the DTD and the meta-DTD. An opaque element is one that behaves normally with respect to validation against its content model.

NOTE 473 Both the element and its children must satisfy the element's parent's current model group without changing the state of content model validation with respect to it. In other words, the content will be validated against the element's parent's content model as though the element itself did not exist.

```
                      <!-- Unique Identifier -->
<!attlist
-- id --          -- Unique identifier --
                  -- Clause: A.5.2 --
   #ALL

   id             -- Unique identifier --
      ID
      #IMPLIED    -- Default: none --
>

                   <!-- Element Type Full Name -->
<!attlist
-- etfullnm --    -- Element type full name --
                  -- Clause: A.5.2 --
   #ALL

   etfullnm       -- Element type full name --
     CDATA
     #IMPLIED     -- Default: generic identifier --
>

                           <!-- Opacity -->
<![ %opacity; [
<!attlist
-- opacity --     -- Opacity --
                  -- Clause: A.5.2 --
   #ALL

   opacity        -- Opacity --
                  -- Transparent or opaque element --
      (transpar|opaque)
      opaque      -- Constant --
>
]]><!-- opacity -->
```

A.5.3 Data Attributes for Elements (DAFE)

In SGML, elements may specify a data content notation to which their content must conform through the use of an attribute with a value prescription of “NOTATION”. Notations may declare data attributes that can be exhibited by the data entities in those notations. However, because elements have their own attribute name space, SGML does not provide a way to specify data attributes on elements when they conform to a specific notation. This limits the utility of notations for elements.

The **data attributes for elements** (*DAFE*) facility of the General Architecture overcomes this problem by providing a means for interpreting element attributes as data attributes for elements that have an associated notation. By using this facility, an element's attributes can be derived from the data attributes of its data content notation, in addition to being derived from base architectures.

When the DAFE facility is in use, elements with a data content notation are treated normally, with one addition: any attribute of the element that has the same name as a data attribute of the notation is assumed to be that data attribute. Its value must satisfy the declared value of both the data attribute definition and the element attribute definition.

NOTE 474 Thus, notations are used much as architectural forms are used. However, the notations are not “architectural forms”, they are notations in the client document. The use of the DAFE facility is reflected in the General Architecture's enhanced SGML document grove, not in a separate architectural grove. In particular, element nodes in the enhanced SGML document grove may exhibit a “data attributes” property in addition to an “attributes” property (see *A.7 SGML Property Set* for details).

A.5.3.1 Data control attributes

The “NotNames” and “NotSupr” attributes shown here are analogous to the ArcNames (see *A.3.5.2 Architectural attribute renamer*) and ArcSupr (*A.3.5.3 Architecture suppressor attribute*) attributes, respectively.

The NotNames attribute maps architecture notation attribute names to substitute client attribute names or element content. When the substitute name is “#DEFAULT”, the architectural attribute is mapped to its architectural default value and cannot be specified by the client element.

Using NotNames, syntactic data content conforming to a notation can be entered as an attribute value by specifying the notation data attribute name as “#NOTCONT” and naming the attribute in whose value it will occur. This must be done if “#CONTENT” is named as a substitute name for another attribute. If done and “#CONTENT” is not named as a substitute name, the content of the element is not governed by the notation.

A substitute name may be followed by one or more triples of tokens each consisting of #MAPTOKEN followed by two name tokens. These specify user substitutes for architecture-defined tokens occurring in the value of this attribute. If a token in a value specified for this attribute in the client document is equal to the second token, then it will be replaced by the first token. If #MAPTOKEN is specified for an architecture-defined attribute whose declared value is CDATA then the value of the attribute shall be tokenized before tokens are substituted.

The **data attributes for elements suppressor** (*NotSupr*) attribute suppresses notation-specific processing of notation attributes for elements. The value “sNotAll” suppresses all notation-specific processing for the element and its descendant elements. The value “sNotForm” suppresses notation-specific processing for the element and its descendant elements, except for processing of the NotSupr attribute itself. The value “sNotNone” turns off suppression of notation-specific processing. If notation-specific processing is suppressed for an element because NotSupr occurs on an ancestor, the element will receive only normal SGML processing. Its element attributes will not be recognized as data attributes.

```
                <!-- Data attributes for elements -->
<![ %dafe; [
<!Attlist
-- dafe --        -- Data attributes for elements --
```

```
                  -- Clause: A.5.3.1 --
   #ALL

   NotNames       -- Data attributes for elements renamer --
                  -- Defines user names for data attributes --
      CDATA       -- Lextype: ((NAME,(ATTORCON|"#DEFAULT"),
                                ("#MAPTOKEN",NMTOKEN,NMTOKEN)*)|
                               ("#NOTCONT",ATTNAME))* --
                  -- Constraint: a given ATTNAME, NAME, #CONTENT, or
             #NOTCONT can occur only once --
                  -- Constraint: data attribute name precedes user
                     name --
      #IMPLIED    -- Constant --
                -- Default: no renaming --

   NotSupr        -- Data attributes for elements suppressor --
                  -- Suppress data attribute for elements
                     processing --
      (sNotAll|sNotForm|sNotNone)
      #IMPLIED    -- Default: inherited --
>
]]><!-- dafe -->
```

A.5.4 Lexical types

A “lexical type” is a defined pattern of characters or tokens to which a character string or token sequence, respectively, conforms. A definition of a lexical type is called a “lexical model”. Data that satisfies a test for conformance to a lexical model is called a “hit”. The definition of lexical types is defined in *A.2 Lexical Type Definition Requirements (LTDR)*.

The attribute form *lextype* consists of an attribute that allows an application to require that attributes and/or content must conform to a lexical type.

The attribute **lexical types** (*lextype*) pairs attributes and/or content with the names of lexical types. Normal SGML parsing of the character data occurs prior to lexical parsing.

NOTE 475 The “lextype” and “ulextype” conventional comments perform the function of lextype for some attribute value and content strings defined by the HyTime language. Care should be taken when defining lextype attributes that constrain such strings to ensure that the constraints of the lextype and ulextype comments can still be satisfied.

```
                  <!-- Lexical Typing Attribute -->
<![ %lextype; [
<!attlist
-- lextype --     -- Lexical typing attribute --
                  -- Clause: A.5.4 --
   #ALL

   lextype        -- Lexical types --
                  -- Lexical types of attribute values or character
                     data content --
      CDATA       -- Lextype: (ATTORCON,NAME)* --
                  -- Constraint: a given ATTNAME or #CONTENT can occur
                     only once --
      #IMPLIED    -- Constant --
                  -- Default: none --
>
]]><!-- lextype -->
```

A.5.5 ID immediate referent type control

The attribute form *ireftype* allows applications to control the immediate referent of ID references.

An ID reference is an attribute whose declared value prescription is IDREF or IDREFS, or an attribute or data content that is declared to contain ID references by a lextype constraint. Attribute ID references are specified by the attribute names, and content ID references by the reserved name “#CONTENT”. The reserved name “#ALL” means “all ID references in the attribute values and data content of this element type”.

The attribute **ID immediate referent element type** (*ireftype*) associates names of referential attributes (or #CONTENT) with element types to which their immediate targets must conform. The element types can be specified as a single GI, possibly followed by an occurrence indicator; a model group of GIs; or the reserved word #ANY, which signifies that any target is acceptable for the corresponding attribute. If the irefmodel option of the General Architecture is supported, model groups can be any valid SGML model group. When the list of element types is not a repeating OR group or when it is a single GI without an occurrence indicator, the reference is limited to a single instance of one of the named types. If the refmodel option is not supported, model groups are limited to single (possibly repeatable) GIs or repeating or non-repeating OR groups of GIs.

NOTE 476 In other words, a simple list of possible target GIs, possibly repeating.

The constraints associated with #ALL can be overridden for particular referential attributes by specifying element types for them individually.

The referenced element type constraints apply to the immediate target of the reference. If the reference is to a location path, the constraints apply to the first location address in the path, not the objects ultimately addressed by the location path.

```
             <!-- ID Immediate Referent Element Type -->
<![ %ireftype; [
<!attlist
-- ireftype --    -- ID immediate referent element type --
                  -- Clause: A.5.5 --
   #ALL

   ireftype       -- ID immediate referent element type --
      CDATA       -- Lextype: (("#ALL",(GI|modelgroup|"#ANY"))?,
                              (ATTORCON,(GI|modelgroup|"#ANY"))*) --
                  -- Constraint: a given ATTNAME or #CONTENT can occur
                     only once; types apply to immediate object of
                     address. --
                  -- Constraint: model groups limited to repeating
                     or non-repeating OR groups if irefmodel option not
                     supported. Tokens in model groups must be GIs. --
      "#ALL #ANY" -- Constant --
>
]]><!-- ireftype -->
```

A.5.6 Default value list

A “default value list” is a means by which an application designer can control the assignment of attribute values in a manner that reflects the element structure of the document.

A.5.6.1 Default value list attributes

The attribute form *dvlatt* allows default attribute values to be associated with an element.

The attribute **subelement default value list** (*subdvl*) specifies default values for impliable attributes that will apply for subelements of the element for which the attribute is specified. The attribute is recursive (that is, default values can be specified for subdvl itself). The attribute value is a list of IDs of default value list elements.

The attribute **sibling default value list** (*sibdvl*) specifies default values for impliable attributes that will apply in the document instance for younger siblings of the element for which the attribute is specified. The attribute is recursive (that is, default values can be specified for sibdvl itself). The attribute value is a list of IDs of default value list elements.

The attribute **self default value list** (*selfdvl*) specifies default values for impliable attributes that will apply in the document instance for the element for which the attribute is specified. The attribute value is a list of IDs of default value list elements.

NOTE 477 The same default value list can be applied to an element, its subelements, and/or its siblings by naming it as the self default value list, the subelement default value list, and/or the sibling default value list.

Default values specified later in the element structure override earlier ones. Specifically, a subdvl will override the subdvl of an ancestor. A sibdvl will override a subdvl, and will also override a sibdvl of an older sibling.

NOTE 478 In other words, at the end of the elder sibling that specifies a sibdvl, the sibdvl is effectively treated as the subdvl of its parent. That is, there is no stacking of sibdvl's over one another, or over the parent subdvl (if any); the parent subdvl is simply replaced by the latest sibdvl at the point where it takes effect.

In addition, if a default value defined in a default value list is defined as a default-setting attribute then the default value will be reset to the value specified on the nearest element of the type controlled by the default value list (ancestors for subelement default value lists, siblings for sibling default value lists).

```
                <!-- Default Value List Attributes -->
<![ %dvlist; [
<!attlist
-- dvlatt --      -- Default value list attributes --
                  -- Clause: A.5.6.1 --
   #ALL

   subdvl         -- Subelement impliable attribute value defaults --
      IDREFS      -- Reference --
                  -- Reftype: dvlist+ --
                  -- Note: Cannot be indirect --
      #IMPLIED    -- Default: none --

   sibdvl         -- Sibling impliable attribute value defaults --
      IDREFS      -- Reference --
                  -- Reftype: dvlist+ --
                  -- Note: Cannot be indirect --
      #IMPLIED    -- Default: none --

   selfdvl        -- Self impliable attribute value defaults --
      IDREFS      -- Reference --
                  -- Reftype: dvlist+ --
                  -- Note: Cannot be indirect --
      #IMPLIED    -- Default: none --
>
]]><!-- dvlist -->
```

A.5.6.2 Default value list element

The element form **default value list** (*dvlist*) contains an attribute specification list. The attribute values are evaluated only when actually used as a default.

The attribute **default value element types** (*dvgi*) specifies the generic identifiers of the element types whose default values are being defined. If not specified, the default value list is applied to all elements within the scope of the default value list. If the keyword “#ALL” is specified, the default value list applies globally to all elements in the document (if no GIs are specified) or to all elements of the types specified.

NOTE 479 Default value lists for which “#ALL” is specified are always processed, whether or not they are referenced by dvlatt attributes.

NOTE 480 Global default value lists can be used as a means of establishing an initial set of default values for all impliable attributes.

The attribute **preempted attributes** (*preatts*) identifies attributes whose values are to be preempted. An attempt to specify such an attribute while the preempted value is in effect is *not* a reportable architectural error, but it has no effect.

NOTE 481 This rule applies to attempts to preempt an already preempted attribute.

NOTE 482 A preempted attribute is similar to an SGML fixed attribute in that no other value can be specified for it. However, it is an error to attempt to respecify a fixed attribute, while it is not an error to attempt to respecify a preempted attribute.

NOTE 483 By preempting the subdvl, sibdvl, and selfdvl attributes, an application designer can prevent the defaults from being changed.

The attribute **default-setting attributes** (*defatts*) names those attributes in the default value list whose values are to become the new default if specified for elements within the scope of the active default value list. In this case, “specified” also means specified for a selfdvl. Default-setting attributes used for subelement default values lists only propagate down the element hierarchy, they do not propagate from elder sibling to younger sibling (unless the same or equivalent default value list is also used as a sibling default value list).

NOTE 484 Application designers can define fixed default value lists by mapping the dvlist-form element's content to an attribute and requiring the element in context.

```
                     <!-- Default Value List -->
<![ %dvlist; [
<!element
   dvlist         -- Default value list --
                  -- Clause: A.5.6.2 --
   - O
   (#PCDATA)      -- Ulextype: attspecs --

-- Attributes: dvlist --
-- CommonAttributes: dafe, dvlist, etfullnm, id, ireftype,
   lextype, opacity --
-- Referrers: dvlatt:selfdvl, dvlatt:sibdvl, dvlatt:subdvl --
>
<!attlist
   dvlist         -- Default value list --
                  -- Clause: A.5.6.2 --

   id             -- Unique identifier --
      ID
      #REQUIRED
   dvgi           -- Default value element types --
                  -- Applies to all elements if omitted --
      CDATA       -- Lextype: (GI+|(#ALL,GI*)) --
      #IMPLIED    -- Default: all elements --

   preatts        -- Attributes whose values are to be preempted --
      NAMES       -- Constraint: must be in dvlist content --
      #IMPLIED    -- Default: none --

   defatts        -- Attributes whose values become the default value
                     when specified --
      NAMES       -- Constraint: must be in dvlist content --
      #IMPLIED    -- Default: no replaceable defaults --
>
]]><!-- dvlist -->
```

A.5.7 Data attributes

SGML allows an attribute definition list to be declared for a data content notation. The attributes on such a list are called “data attributes”. They are specified on entity declarations for external data entities that use the notation.

A.5.7.1 Common data attributes

The attribute forms **included**, **altreps**, and **superdcn**, provide common data attributes that an application can use to interpret the data of the entity. These attributes can be used individually in attribute list definitions; each is a separate facility.

To be recognized as General Architecture data attributes, client notations that use the common data attributes must be derived, directly or indirectly, from the General Architecture GABrid notation form.

The attribute **included entities** (*included*) identifies entities that are included by reference from within data in a notation other than SGML (and therefore not recognized by the SGML parser). This information can be useful when moving or packing the data, as it allows the included entities to be found without knowing the notation of the data.

NOTE 485 A typical use of included entities referenced from notation data is a re-used object, such as symbols included by reference in a CGM level 4 graphic. Several entities could share these symbol definitions without duplicating them.

Another example is to support object-oriented programming by identifying methods that an application can invoke to access encapsulated data in the entity.

The attribute **alternative representations** (*altreps*) identifies equivalent representations of the entity, possibly using different data content notations.

NOTE 486 Alternative representations are not necessarily identical equivalents. They are typically used in fallback situations, when the preferred representation cannot be accessed or processed for some reason. However, the representations are presumed to be semantically equivalent from an information content standpoint, e.g., all the alternate representations of a graphic are more or less the same picture.

The attribute **notation derivation source** (*superdcn*) identifies a data content notation from which the current notation is derived.

NOTE 487 The superdcn attribute defines a derivation hierarchy within a single client document or architecture. Notations can also be derived from architectural notation forms. These two derivation hierarchies are independent of each other.

```
            <!-- Entities included from notation data -->
<![ %included; [
<!attlist #NOTATION
-- included --    -- Entities included from notation data --
                  -- Clause: A.5.7.1 --
   #ALL

   included       -- Entities included from notation data --
      CDATA       -- Lextype: ENTITIES --
      #IMPLIED    -- Default: no included entities --
>
]]><!-- included -->

                  <!-- Alternate Representations -->
<![ %altreps; [
<!attlist #NOTATION
-- altreps --     -- Alternate representations --
                  -- Clause: A.5.7.1 --
   #ALL

   altreps        -- Alternate representations --
                  -- Alternative representations of this entity --
      CDATA       -- Lextype: ENTITIES --
      #IMPLIED    -- Default: none --
>
]]><!-- altreps -->

                 <!-- Notation Derivation Source -->
<![ %superdcn; [
<!attlist #NOTATION
-- superdcn --    -- Notation derivation source --
                  -- Clause: A.5.7.1 --
   #ALL

   superdcn       -- Notation derivation source --
                  -- Notation on which this one is based --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: none --
>
]]><!-- superdcn -->
```

A.5.8 Conformance

Documents, applications, and systems may all conform to this Annex.

A.5.8.1 Conforming General Architecture document

If a General Architecture document complies with all provisions of this Annex and is a conforming SGML document as defined in ISO 8879, it is a conforming General Architecture document.

A.5.8.1.1 Minimal General Architecture document

A conforming General Architecture document is a minimal General Architecture document if its General Architecture declarations are as follows:

```
<!Notation
   GenArc         -- General Architecture --
```

```
                  -- A base architecture used in conformance with the
                     Architectural Form Definition Requirements of
                     International Standard ISO/IEC 10744. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           General Architecture//EN"
>
<!Attlist #NOTATION
   GenArc         -- General Architecture --

   ArcFormA NAME     GAForm
   ArcNamrA NAME     GANames
   ArcSuprA NAME     GASupr
   ArcIgnDA NAME     GAIgnD
   ArcDocF  NAME     GADoc
   ArcDTD   CDATA    "GenArc"

   -- NAMELEN must be at least 9 because the General Architecture
      meta-DTD uses 8-character parameter entity names. --
   ArcQuant CDATA    #FIXED "NAMELEN 9"

   ArcDataF NAME     GABrid
   ArcBridF NAME     GABrid
   ArcAuto  (ArcAuto|nArcAuto) ArcAuto
   ArcOptSA NAMES    #FIXED "commatts dcnatts"
>
```

A.5.8.2 Conforming General Architecture application

If a General Architecture application meets the requirements of this sub-clause and is a conforming SGML application as defined in ISO 8879, it is a conforming General Architecture application.

A.5.8.2.1 Application conventions

A conforming General Architecture application's conventions can affect only areas that are left open to specification by applications.

NOTE 488 Some examples are: names of element types conforming to General Architecture architectural forms, and substitute attribute names.

A.5.8.2.2 Conformance of documents

A conforming General Architecture application shall require its documents to be conforming General Architecture documents, and shall not prohibit any markup that this International Standard would allow in such documents.

NOTE 489 For example, an application markup convention could recommend that only certain minimization functions be used, but could not prohibit the use of other functions if they are allowed by the formal specification.

A.5.8.2.3 Conformance of documentation

A conforming General Architecture application's documentation shall meet the requirements of this International Standard (see *11.5 Documentation requirements*).

A.5.8.3 Conforming General Architecture system

If a General Architecture system meets the requirements of this sub-clause and is a conforming SGML system as defined in ISO 8879, it is a conforming General Architecture system.

A.5.8.3.1 Conformance of documentation

A conforming General Architecture system's documentation shall meet the requirements of this International Standard (see *11.5 Documentation requirements*).

A.5.8.3.2 Conformance to General Architecture system declaration

A conforming General Architecture system shall be capable of processing any conforming General Architecture document that is not inconsistent with the system's General Architecture system declaration (see *A.5.9 General Architecture system declaration*). A conforming General Architecture system shall report every unsupported General Architecture facility used in any General Architecture document it processes.

NOTE 490 A system's inability to process data content notations that are not defined in this International Standard does not affect whether it is a conforming General Architecture system.

A.5.8.3.3 Support for minimum General Architecture documents

A conforming General Architecture system shall be capable of processing a minimal General Architecture document.

A.5.8.3.4 Application conventions

A conforming General Architecture system shall not enforce application conventions as though they were requirements of this International Standard.

NOTE 491 Warnings of the violation of application conventions can be given, but they must be distinguished from reports of General Architecture errors.

A.5.8.4 Validating General Architecture engine

If a General Architecture engine in a conforming General Architecture system meets the requirements of this sub-clause, it is a validating General Architecture engine.

NOTE 492 A conforming General Architecture system need not have a validating General Architecture engine. Implementors can therefore decide whether to incur the overhead of validation in a given system.

A.5.8.4.1 Error recognition

A validating General Architecture engine shall find and report a reportable General Architecture error if one exists, and shall not report an error when none exists.

A validating General Architecture engine can optionally report other errors.

NOTE 493 This International Standard does not specify how a General Architecture error should be handled, beyond the requirement for reporting it. In particular, it does not state whether the erroneous information should be treated as data, and/or whether an attempt should be made to continue processing after an error is found.

NOTE 494 This International Standard does not prohibit a validating General Architecture engine from reporting an error that this International Standard considers non-reportable if the engine is capable of doing so. Neither does it prohibit an engine from recovering from such errors, nor does it require an engine to report them when it is not performing validation.

A validating General Architecture engine may warn of conditions that are potentially, but not necessarily, errors.

NOTE 495 General Architecture architectural forms do not constrain the construction of document type definitions, only document instances. However, a validating General Architecture engine can optionally report DTD constructs that would prevent the creation of a valid conforming instance, or that would allow the creation of a nonconforming instance.

A.5.8.4.2 Identification of General Architecture messages

Reports of General Architecture errors, including optional reports, shall be identified as General Architecture messages in such a manner as to distinguish them clearly from all other messages, including warnings of potential General Architecture errors.

A.5.8.4.3 Content of General Architecture messages

A report of a General Architecture error, including an optional report, shall state the nature and location of the error in sufficient detail to permit its correction.

NOTE 496 This requirement is worded to allow implementors maximum flexibility to meet their user and system requirements.

A.5.8.5 Standard identification

Standard identification shall be in the natural language of the documentation.

Standard identification text shall be displayed prominently:

a) in a prominent location in the front matter of all publications (normally the title page and cover page);

b) on all identifying display screens of HyTime programs; and

c) in all promotional and training material.

For applications, the identification text is:

```
A General Architecture application conforming to
the General Architecture Annex of
International Standard ISO/IEC 10744 --
Hypermedia/Time-based Structuring Language
```

For systems, the identification text is:

```
A General Architecture system conforming to
the General Architecture Annex of
International Standard ISO/IEC 10744 --
Hypermedia/Time-based Structuring Language
```

The documentation for a conforming General Architecture system shall include a General Architecture system declaration (see *A.5.9 General Architecture system declaration*).

A.5.9 General Architecture system declaration

A General Architecture system declaration specifies the version of the General Architecture and all of the optional facilities that a General Architecture system can support. It is represented by a set of General Architecture support declarations in the reference concrete syntax. There must be one option attribute for each supported facility, within which there can be no duplication of option names.

The system declaration for a system that supports all of the optional facilities of General Architecture is:

```
<!Notation
   GenArc         -- General Architecture --
                  -- A base architecture used in conformance with the
                     Architectural Form Definition Requirements of
                     International Standard ISO/IEC 10744. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           General Architecture//EN"
>
<!Attlist #NOTATION
   GenArc         -- General Architecture --

   commatts       -- Common element attribute options --
      CDATA       -- Lextype: csname+ --
                  -- Constraint: lists parameter entities in meta-DTD
                     to be set to "INCLUDE" --
      "dafe dvlist ireftype HyLex HyOrd lextype opacity REGEX"

   dcnatts        -- Common data attribute options --
      CDATA       -- Lextype: csname+ --
                  -- Constraint: lists parameter entities in meta-DTD
                     to be set to "INCLUDE" --
      "altreps included superdcn"
>
```

A.6 Formal System Identifier Definition Requirements (FSIDR)

This annex states the requirements for the formal definition of notations used in system identifiers. System identifiers are markup declaration parameters that specify access to the storage objects in which entities are stored. Access is provided by a “storage manager” (SM) such as a file system, data base, network, or main memory manager. Objects may be stored individually, or as part of larger storage objects, called “containers” or “archives”, with a defined format for multiple-object storage (e.g. MIME, TAR, etc.). Access may involve auxiliary processes, such as transformation from octet sequences to the internal representation of characters, record boundary recognition, and other processes required to present storage objects to the SGML parser as entities.

An SGML system provides access to storage managers for its documents whether or not the facilities defined in this annex are used. This annex therefore serves two distinct purposes:

a) It specifies a standardized structure for system identifiers, known as a “Formal System Identifier” (FSI). An FSI can support arbitrary mappings between entities and storage objects, including one-to-many, many-to-one, distributed storage, and containers.

b) It provides a means of formally declaring the storage management facilities of the system on which the document was created. Those facilities are declared in the system's “FSI definition document” (fsidd) and, in any case, are available whether or not so declared. Therefore, a system that supports Formal System Identifiers can use them outside of documents for such purposes as identifying an SGML document entity to be processed, or maintaining a catalog that maps public identifiers to Formal System Identifiers.

This annex is specified as an architecture conforming to the Architectural Forms Definition Requirements annex of this International Standard. This architecture defines a specialized syntax and associated semantics for system identifiers as well as a set of notation forms for storage managers defined in formal system identifier definition documents. In addition to the base storage manager notation forms, this annex also defines a “starter set” of storage managers derived from the base forms. Applications and systems may provide support for formal system identifiers without otherwise supporting architectures or architectural processing as only the fsidd documents are actually derived from the FSIDR architecture.

A.6.1 System identifiers

An entity is a virtual storage object. A system identifier parameter of a markup declaration can be used to map an entity onto one or more real storage objects (or portions thereof), and to specify processes to be performed in the course of accessing the object as an entity. The format of a system identifier is normally system-specific (an “informal system identifier”). However, when access to storage is specified in the manner described in this annex, the system identifier is called a “formal system identifier” (FSI).

NOTE 497 A given system identifier is either informal or formal. Both kinds can occur in the same document.

A formal system identifier consists of one or more “storage object specifications (SOS)”, each of which identifies a storage object. The storage objects are concatenated in the order specified to comprise the storage of the entity. When the “altsos” option is supported, the FSI can contain alternative SOS sequences that identify alternative (that is, duplicate) storage locations for the identical objects; the system is free to choose the most convenient.

A.6.1.1 Storage object specification (SOS)

The format of an SOS resembles an element, in that it consists of a start-tag followed by content, followed optionally by an end-tag or empty end-tag.

In the SOS tag, the name appearing as the generic identifier is that of the storage manager (SMName). It is the name of a “storage manager notation” that was identified as such by a declaration. There can also be an attribute specification list, consisting of attributes defined for the storage manager notation. These SOS attributes serve as parameters that govern the access to the storage object.

The content of an SOS is known as the “storage object identifier” (SOI). Its syntax and semantics depend on the individual storage manager.

A.6.1.2 Informal system identifiers

A system identifier is recognized as an FSI only if it begins with an SOS tag. Informal system identifiers can be used in the same document as FSIs as long as storage manager names are chosen so that informal system identifiers don't appear to begin with an SOS tag.

A system FSI definition document (fsidd) should define how informal system identifiers are interpreted. Typically, they are treated as SOIs of a particular SM, and given the default values of the SM's attributes.

A.6.1.3 Entity usage attributes

An entity can have attributes provided by the notation in which the entity body is represented. These attributes are called “data attributes”; they describe aspects of interpreting the notation.

The storage objects in which an entity is stored can also have attributes. They are provided by the storage managers and describe aspects of physically accessing the objects and converting them to the form expected by the SGML parser or other notation interpreter.

An entity can have additional properties of its own that are independent of both the notation and the storage manager. They are called “entity usage attributes” and they describe aspects of the use of the entity, such as authorizations, modification levels, and alternative representations.

A.6.2 Auxiliary processes

Several auxiliary processes may be required to convert a newly-created entity into the form in which it will be stored. Conversely, auxiliary processes may be needed to convert the octets of a storage object to the internal representation of characters seen by an SGML parser.

The SGML language is designed so that SGML documents can be stored in the same way that a system's ordinary text files are stored, thereby allowing access and processing with normal text processing tools in case SGML-aware tools are unavailable or are deficient in some respect.

As a result, after an entity is created or modified a number of processes can take place as it is stored. First, if the entity is not to be stored in a single storage object, it is divided into as many portions as there are to be storage objects. The storage objects can be in different storage systems (that is, under different storage managers). For each portion, the following steps may be performed:

a) Record boundaries are converted to the storage system form of line endings for text files (for example, carriage returns, line feeds, or both).

b) The representation of characters used in storage objects may be different from that used internally by the SGML parser.

   NOTE 498 With large character sets it is often the case that characters are represented internally by a fixed-length sequence of octets, but externally by a variable-length sequence of octets.

c) In this case the internal representation of characters must be converted to the external representation as a sequence of storage octets.

d) The location that the entity body is to occupy in the storage object is determined (if it is not the entire object). The entity body can occupy one or more extents in the storage object.

e) The storage object may be encrypted.

f) The storage object may be compressed.

g) The storage object may be “sealed” by calculating a check number that will no longer be valid if the storage object is modified.

When an entity is accessed, the entity manager invokes the storage manager for each storage object specification. For each storage object, the following steps may be performed:

a) The integrity of the storage object (if sealed) is verified.

b) The storage object is decompressed (if it was compressed).

c) The storage object is decrypted (if it was encrypted).

d) The extents of the storage object that are occupied by the entity are located and concatenated into a single portion of stored data.

e) The octet sequences are converted to the internal representation of characters used by the parser. This conversion depends on the encoding of the storage of the storage object and the internal representation of characters used by the system; it may be a null operation.

f) Record boundaries are recognized and converted to SGML RE and RS characters.

NOTE 499 Although the auxiliary processing is described sequentially for clarity, an implementation can perform the processes in parallel and in any order as long as identical results are achieved.

A.6.3 Containers

The most common storage managers are the file systems of operating systems and networks. These have the property that the objects stored by them are not stored within some other object. In contrast, several storage managers, such as archivers and data base managers, store objects within a larger object. Typically, those objects (the “archives” or “data bases”) are stored as files by a file system, although in some cases they could be stored within a larger archive or data base.

In the context of formal system identifiers, storage managers that store objects within larger objects are known as “container storage managers”. The notation for a container storage manager's SOI is known as a “container notation”, derived from the FSIDR container notation form.

A container is an entity whose storage is partitioned so that the bodies of other entities (“contained objects”) can be kept in it. The locations of the contained objects are specified by their entity declarations. The entity declarations serve as entries in a “table of contents” or “directory” of the container.

NOTE 500 Container entities provide a storage organization that applications may take advantage of to avoid redundant descriptor information. Containers may also facilitate interleaving and other techniques that optimize access to multimedia data.

When a container SM is used, the characters of the container entity are treated as the storage octets of the contained entities.

NOTE 501 When a query language (such as SQL or SDQL) is used as a storage manager notation, only queries that return a single storage object are valid. In the usual case, the query language will be a container notation, with the container entity being the relational table or other query domain and the SOI of the contained entity being the query.

A.6.4 FSI identification facilities

A document indicates the presence of formal system identifiers by using the APPINFO parameter of the SGML declaration and/or the other facilities described in this sub-clause.

In describing these facilities, it is frequently necessary to distinguish between the storage object in an FSI that is under discussion, its associated entity, and the entity and storage object in which the FSI occurs. The following terms are used:

specified storage object
The storage object under discussion. It is specified in an FSI.

declared entity The entity whose declaration contains the specified storage object's FSI, or caused that FSI to be accessed from a catalog.

current storage object
The storage object in which the FSI occurs. If the FSI is in a catalog entry, it is the catalog storage object.

current entity The entity in which the declared entity's declaration occurs.

specified storage manager
The storage manager of the specified storage object.

current storage manager
The storage manager of the current storage object.

A.6.4.1 FSI use of APPINFO parameter

To determine when formal system identifiers are in use, the APPINFO parameter of the SGML declaration is parsed as a space-delimited sequence of tokens. Potential use of one or more storage managers defined in accordance with these requirements is indicated by specifying the keyword “FSIDR” as one such token. The keyword indicates the potential presence in one or more DTDs or LPDs of a formal system identifier (FSI) declaration that conforms to these formal system identifier definition requirements. The use of the APPINFO parameter to indicate the use of formal system identifiers is optional.

NOTE 502 The use of APPINFO serves to declare the use of formal system identifiers within SGML declarations. However, it is sufficient to enable formal system identifier processing to declare the use of formal system identifiers with the FSIDR processing instruction within DOCTYPE declarations.

The format of the token is:

```
FSIDR
```

The token can also specify the name of the FSI declarations in the document's DTDs or LPDs if it is other than “FSIDR”. The format is:

```
FSIDR=FSIUsed
```

where “FSIUsed” is replaced by the actual FSI declaration name.

A.6.4.2 FSI declaration

An FSI declaration identifies one or more storage manager notations used in system identifiers. System identifiers in which such notations occur are known as “formal system identifiers”. There can be only one FSI declaration in a DTD or LPD.

Syntactically, the FSI declaration is a processing instruction (PI), not an SGML markup declaration. In the template below, it is shown in the reference concrete syntax. In use, the SMName-list parameter must be replaced by one or more storage manager names (SMName) declared as notation names, separated by SGML *s* separators (white space). It is a reportable error if the system's fsidd does not declare a notation for each SMName specified.

An optional FSIDefDoc parameter can follow the SMName-list parameter, separated from it by a ts. In use, the external-id in the template is replaced by the identifier for the fsidd that defines the SMs used in this document. The parameter can be omitted when the system's only (or default) fsidd is used.

In the (admittedly unlikely) event that a document contains informal system identifiers that appear to be FSIs, and the smalias option is supported, an SMName can be accompanied by an alias that will be used in FSIs in its place. The format is:

```
SMName=SMNameAlias
```

where “SMNameAlias” is replaced by the actual name to be used for that storage manager in the FSIs in the document.

Following the initial character string of “IS10744 ”, the declaration name is the following character string, up to the first ts separator. The name is always “FSIDR” in meta-DTDs. It should also be “FSIDR” in DTDs or LPDs, but provision is made for changing it in the The name is always “FSIDR” in meta-DTDs. It should also be “FSIDR” in DTDs or LPDs, but provision is made for changing it in the APPINFO parameter of the SGML declaration, if necessary, to avoid the (admittedly unlikely) possibility of conflicts when retrofitting FSIs to a document that already has PIs that begin with “IS10744 FSIDR ”. The declaration name is subject to upper-case substitution if the SGML declaration of the client document specifies general upper-case substitution.

```
              <!-- FSI Storage Managers Declaration -->
         <!-- TEMPLATE FOR PI IN DTD, LPD OR DERIVED META-DTD -->
<?IS10744 FSIDR SMName-list FSIDefDoc="external-id" >
```

A.6.4.3 FSI syntax

A formal system identifier consists of a sequence of one or more storage object specifications (an “SOS sequence”). The characters resulting from the resolution of each SOS are concatenated in the order of the sequence and comprise the characters of the entity.

The reference concrete syntax is used for recognition of SOSs in FSIs, both within and outside of SGML documents.

An SOS start-tag is recognized by the occurrence of a start-tag open delimiter followed immediately by a declared storage manager name (see *A.6.4.2 FSI declaration*) followed either by an SGML “s” separator or a tag-close delimiter.

NOTE 503 SGML numeric character references are recognized in the attribute value literals and content (SOIs) of an SOS. They can be used to avoid false delimiter recognition.

Record starts and record ends are handled in an SOI as they are handled in syntactic content in SGML.

NOTE 504 In practice, this rule means that record starts are deleted from the SOI and that a record end immediately after an SOS start-tag or at the end of an SOI is also deleted.

When the alternative SOS sequence (altsos) option is supported, there can be more than one SOS sequence, separated by vertical bars. An end-tag is required before the vertical bar to avoid ambiguity as to whether the vertical bar is part of the SOI. All the SOS sequences are expected to yield the identical character sequences. The system is free to choose the most convenient.

When the entity usage (entuse) SM is declared (see *A.6.6 Entity usage attribute definitions*), the FSI can optionally begin with an entuse start-tag, which can be used to specify attributes that apply to all of the entity. The entuse start-tag is not part of any alternative SOS sequence; it applies independently of which alternative is selected. There is no corresponding end-tag for the entuse start-tag. The entuse notation is not a storage manager; it is used solely for the purpose of declaring and specifying attributes.

NOTE 505 The structure of an FSI can be summarized as:

[35] fsi =
    entuse start-tag?,
    sos sequence,
   (**or**,
     sos sequence)*,

NOTE 506

[36] sos sequence =
    sos+

NOTE 507

```
[37] sos =
    sos start-tag,
    soi,
   (  sos end-tag|
     (  etago,
        tagc))?
```

A.6.5 Storage manager attribute definitions

The designer of a storage object specification can optionally associate an attribute definition list declaration with the notion declaration used to identify the storage manager processor. The attributes are used to specify parameters to the storage access, in addition to the storage object identifier.

A starter set of standardized storage manager attributes is defined in these FSIDR requirements.

NOTE 508 Some storage managers might choose to include information represented by one or more of these attributes in its SOI syntax, in which case the attribute should not be defined for that SM.

NOTE 509 As with all SGML notations, it is possible to declare “data attributes” that can be passed as parameters to the notation handler. For auxiliary process notations, however, only the default value of the attributes can be specified; there is no way to specify attributes for individual uses of the notation. Therefore, if alternative sets of parameters are needed for a particular auxiliary process, a different notation should be declared for each set, but with the same external identifier. The several names will then invoke the identical auxiliary process, but with varying parameter sets.

A.6.5.1 Record-related attributes

The attribute form **record-processing attributes** (*records*) consists of attributes that control how the entity data is interpreted as records or lines.

The attribute **record boundary indicator** (*records*) identifies the character or characters that are interpreted as a record boundary in an SGML entity.

The keyword “ASIS” means that no attempt will be made to interpret the input as consisting of records. This keyword is used either because the entity body is to be read as blocks instead of characters (e.g. for data entities), or because the storage object already contains the record boundary characters required by the concrete syntax.

If it is known that one of the four line terminator conventions (“LF”, “CR”, “CRLF” and “LFCR”) is used, it can be specified directly. Otherwise, “FIND” can be specified and the first of the four found in the storage object (if any) will be treated as the record boundary indicator for the rest of that storage object (unless the system uses RMS). If none is found, it is the equivalent of specifying “ASIS”.

The keyword “RMS” stands for “Record Management System”, wherein the storage manager recognizes record boundaries by means other than a character sequence (typically, it knows the length of each record).

When records are recognized in a storage object, a record start is inserted at the beginning of each record, and a record end at the end of each record. If there is a partial record (a record that doesn't end with the line terminator) at the end of the entity, then a record start will be inserted before it but no record end will be inserted after it.

NOTE 510 The literal SM can be used to insert a trailing record end, if desired.

The attribute **record tracking** (*tracking*) specifies whether the entity manager must include record count information in messages.

NOTE 511 Some implementations could improve performance by not tracking records, particularly in very large storage objects.

```
                <!-- Record-processing Attributes -->
<!attlist #NOTATION
-- records --     -- Record-processing attributes --
                  -- Clause: A.6.5.1 --
   #ALL

   records        -- Record boundary recognition --
      (asis|crlf|cr|find|lfcr|lf|rms)
      #IMPLIED    -- Default: find, except "asis" for NDATA entities
                     and those whose FSIs are in storage objects with
                     records=asis. --

   tracking       -- Record boundary tracking in messages --
                  -- Constraint: SGML entities only --
      (track|notrack)
      track
>
```

A.6.5.2 Encoding-related attributes

The attribute form **entity encoding specification** (*encoding*) consists of attributes that specify the encoding method used for the entity.

When an entity is stored, the characters of the entity must be converted from their internal representation to a representation as a sequence of storage octets. Since the internal representation of characters is system-dependent, this conversion is not specified directly. Instead a mapping from characters to octet sequences, known as an “encoding”, is specified. The storage manager determines the conversion algorithm using the specified encoding together with its inherent knowledge of the internal representation used for characters.

The encoding can be specified in one of two ways:

— It may be specified completely using the *encoding* attribute. When so specified, the encoding is independent of the document character set of the document in which the entity is used.

— It may be specified partially by the attribute **bit combination transformation format** (*bctf*), which identifies an algorithm for mapping from fixed-size bit combinations to the octet sequences of a storage object. In this case the encoding is the combination of

- the mapping from characters to fixed-size bit combinations determined by the document character set, with
- the mapping from fixed-size bit combinations to octet sequences specified by the BCTF.

When an entity is read, the inverse of the encoding mapping is used.

NOTE 512 When a storage object is in a container its “octets” are actually the characters of the container entity. In such cases, the encoding or bctf attributes are normally specified for either the container or the contained object, but not both.

```
                <!-- Entity Encoding Specification -->
<!attlist #NOTATION
-- encoding --    -- Entity encoding specification --
                  -- Clause: A.6.5.2 --
   #ALL

   encoding       -- Encoding --
                  -- Constraint: at most one of encoding and bctf may
                     be specified --
                  -- Constraint: SGML, CDATA, SDATA entities only --
      NAME        -- Constraint: registered value notation --
      #IMPLIED    -- Default: same unless bctf is specified --

   bctf           -- Bit combination transformation format --
                  -- Constraint: SGML, CDATA, SDATA entities only --
                  -- Constraint: at most one of encoding and bctf may
                     be specified --
      NAME        -- Constraint: registered value notation --
      #IMPLIED    -- Default: storage object does not have
                     document-character-set-dependent character set --
>
```

A.6.5.2.1 Encoding notations

A starter set of notations for encodings is defined in this International Standard. The notations are:

ucs-2 This encoding is the two-octet BMP form of coded representation of the Universal Multiple-Octet Coded Character Set defined in ISO/IEC 10646.

ucs-4 This encoding is the four-octet canonical form of coded representation of the Universal Multiple-Octet Coded Character Set defined in ISO/IEC 10646.

utf-8 This encoding is the UCS Transformation Format 8 defined in Annex P of Amendment 1 to PDAM 1 of ISO/IEC 10646-1:1993. This encodes a character in the repertoire of ISO/IEC 10646 using between 1 and 6 octets.

utf-16 This encoding is the UCS Transformation Format 16 defined in Annex Q of Amendment 1 to ISO/IEC 10646-1:1993.

utf-7 This encoding is the UCS Transformation Format 7 encoding defined by IETF RFC 1642. This can be used to encode characters in the Basic Multilingual Plane of ISO/IEC 10646.

unicode This represents each character in the Basic Multilingual Plane of ISO/IEC 10646 by two octets. The bytes representing the entire storage object may be preceded by a pair of bytes representing the byte order mark character (0xFEFF). The bytes representing each bit combination are in the system byte order, unless the byte order mark character is present, in which case the order of its bytes determines the byte order. When the storage object is read, any byte order mark character is discarded.

euc-jp This encoding is the Extended UNIX Code Packed Format for the Japanese registered Internet character set.

sjis This is the Shift JIS registered Internet character set.

is8859-N where N can be any single digit other than 0. This encodes a character from ISO 8859-N with a single octet.

same The encoding of this storage object is the same as the encoding of the storage object in which the SOS of this storage object is specified.

```
                     <!-- Encoding Notations -->
              <!-- THIS IS A NON-MANDATORY STARTER SET. -->
<!notation
   UCS-2          -- UCS-2 encoding --
                  -- This encoding is the two-octet BMP form of coded
                     representation of the Universal Multiple-Octet
                     Coded Character Set defined in ISO/IEC 10646. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           UCS-2 Encoding//EN"
>
<!notation
   UCS-4          -- UCS-4 encoding --
                  -- This encoding is the four-octet canonical form of
                     coded representation of the Universal
                     Multiple-Octet Coded Character Set defined in
                     ISO/IEC 10646. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           UCS-4 Encoding//EN"
>
<!notation
   UTF-8          -- UTF-8 encoding --
                  -- This encoding is the UCS Transformation Format 8
                     defined in Annex P to PDAM 1 of ISO/IEC
                     10646-1:1993.  This encodes a character in the
                     repertoire of ISO/IEC 10646 using between 1 and 6
                     octets. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           UTF-8 Encoding//EN"
>
<!notation
   UTF-16         -- UTF-16 encoding --
                  -- This encoding is the UCS Transformation Format 16
                     defined in Annex Q of Amendment 1 to ISO/IEC
                     10646-1:1993. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           UTF-16 Encoding//EN"
>
<!notation
   UTF-7          -- UTF-7 encoding --
                  -- This encoding is the UCS Transformation Format 7
                     encoding defined by IETF RFC 1642.  This can be
                     used to encode characters in the Basic
                     Multilingual Plane of ISO/IEC 10646. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           UTF-7 Encoding//EN"
>
<!notation
   UNICODE        -- UNICODE encoding --
                  -- This represents each character in the Basic
                     Multilingual Plane of ISO/IEC 10646 by two
                     octets. The bytes representing the entire storage
                     object may be preceded by a pair of bytes
                     representing the byte order mark character
                     (0xFEFF). The bytes representing each bit
                     combination are in the system byte order, unless
                     the byte order mark character is present, in
                     which case the order of its bytes determines the
                     byte order. When the storage object is read, any
                     byte order mark character is discarded. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           UNICODE Encoding//EN"
>
<!notation
   EUC-JP         -- EUC-JP encoding --
                  -- This encoding is the Extended UNIX Code Packed
                     Format for the Japanese registered Internet
                     character set. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           EUC-JP Encoding//EN"
>
<!notation
   SJIS           -- SJIS encoding --
                  -- This is the Shift JIS registered Internet
                     character set. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           SJIS Encoding//EN"
>
<!notation
   IS8859-1       -- ISO8859-1 encoding --
                  -- This encodes a character from ISO 8859-1 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-1 Encoding//EN"
>
<!notation
   IS8859-2       -- ISO8859-2 encoding --
                  -- This encodes a character from ISO 8859-2 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-2 Encoding//EN"
>
<!notation
   IS8859-3       -- ISO8859-3 encoding --
                  -- This encodes a character from ISO 8859-3 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-3 Encoding//EN"
>
<!notation
   IS8859-4       -- ISO8859-4 encoding --
                  -- This encodes a character from ISO 8859-4 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-4 Encoding//EN"
>
<!notation
```

```
   IS8859-5       -- ISO8859-5 encoding --
                  -- This encodes a character from ISO 8859-5 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-5 Encoding//EN"
>
<!notation
   IS8859-6       -- ISO8859-6 encoding --
                  -- This encodes a character from ISO 8859-6 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-6 Encoding//EN"
>
<!notation
   IS8859-7       -- ISO8859-7 encoding --
                  -- This encodes a character from ISO 8859-7 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-7 Encoding//EN"
>
<!notation
   IS8859-8       -- ISO8859-8 encoding --
                  -- This encodes a character from ISO 8859-8 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-8 Encoding//EN"
>
<!notation
   IS8859-9       -- ISO8859-9 encoding --
                  -- This encodes a character from ISO 8859-9 with a
                     single octet. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           ISO8859-9 Encoding//EN"
>
<!notation
   SAME           -- Same encoding --
                  -- The encoding of this storage object is the same
                     as the encoding of the storage object in which
                     the SOS of this storage object is specified. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR ENCODING
           No change in encoding//EN"
>
```

A.6.5.2.2 BCTF algorithm notations

A starter set of notations for BCTF algorithms is defined in this International Standard. The notations are:

identity Each bit combination is represented by a single octet; this BCTF can be used only for storage objects all of whose bit combinations have a value not exceeding 255.

fixed-2 Each bit combination is represented by exactly 2 octets, with the more significant octet first; this BCTF can be used only for storage objects all of whose bit combinations have a value not exceeding 65,535.

fixed-3 Each bit combination is represented by exactly 3 octets, with a more significant octet preceding any less significant octets; this BCTF can be used only for storage objects all of whose bit combinations have a value not exceeding 16,777,215.

fixed-4 Each bit combination is represented by exactly 4 octets, with a more significant octet preceding any less significant octets; this BCTF can be used only for storage objects all of whose bit combinations have a value not exceeding 4,294,967,295.

```
                 <!-- BCTF Algorithm Notations -->
           <!-- THIS IS A NON-MANDATORY STARTER SET. -->
<!NOTATION identity
   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR BCTF
           IDENTITY BCTF Algorithm//EN"
>
<!NOTATION fixed-2
   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR BCTF
           FIXED-2  BCTF Algorithm//EN"
>
<!NOTATION fixed-3
   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR BCTF
           FIXED-3  BCTF Algorithm//EN"
>
<!NOTATION fixed-4
   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR BCTF
           FIXED-4  BCTF Algorithm//EN"
>
```

A.6.5.3 Common storage manager attributes

The attribute form **common storage manager attributes** (*smcommon*) consists of attributes that control common aspects of storage object access and SOS interpretation.

The attribute **occupied extents** (*extents*) is a marklist notation dimlist that specifies the extents of the storage object occupied by the entity. Multiple dimension specifications can be specified if the entity is segmented and distributed in several locations within the storage object.

NOTE 513 For example, this technique can be used to interleave the bodies of entities that are accessed concurrently.

The attribute **zap end-of-file** (*zapeof*) specifies whether the trailing octet sequence of a storage object should be excluded from the entity when it represents an end-of-file.

NOTE 514 Note that this is different functionality from the extents attribute because with zapeof the user doesn't have to know whether the trailing octets of a particular storage object were in fact the end-of-file control (e.g. the control-Z octet in DOS). Zapeof should normally be specified for DOS (this is the behavior of the standard C libraries with respect to text files).

The attribute **storage manager character reference delimiter** (*smcrd*) is a single character that delimits a reference to a character in the inherent character set of the storage manager (which may be different from the character set of the document in which the FSI occurs). The smcrd is recognized only in an SOI or attribute value, and only if followed by a decimal digit. The smcrd character, together with following decimal digits and an optional semicolon, are replaced by the character with the specified number in the inherent character set of the storage manager.

The attribute **compression information** (*compress*) can be used to identify a compression/decompression process, declared as a notation.

The attribute **encryption information** (*encrypt*) can be used to identify an encryption/decryption process, declared as a notation.

NOTE 515 It may be considered poor security practice to include this information, or even to indicate that the storage object is encrypted.

The attribute **integrity information** (*seal*) can be used to identify a verification process, declared as a notation. One registered value for this attribute is provided in the starter set, md5, the standard form of sealing used for transmission of documents over the Internet.

The attribute **base for relative SOI** (*SOIbase*) allows an alternative relative base to be specified. Its value must be an SOI for the SM of the specified storage object. SOIbase can be declared for any storage manager that allows a relative SOI.

NOTE 516 Using an SOIbase attribute may not be equivalent to prepending a string to the SOI for a storage manager that does searching. For example, with an FSI

```
<osfile soibase=entities>foo.sgm</osfile>
```

NOTE 517 A storage manager might look first for a file entities/foo.sgm and then, if that did not exist, for a file pubtext/foo.sgm, whereas with

```
<osfile>entities/foo.sgm</osfile>
```

NOTE 518 it might look first for a file entities/foo.sgm but then for a file pubtext/entities/foo.sgm.

```
              <!-- Common Storage Manager Attributes -->
<!attlist #NOTATION
-- smcommon --    -- Common storage manager attributes --
                  -- Clause: A.6.5.3 --
   #ALL

   extents        -- Dimensions of occupied extents of object --
                  -- Constraint: applies to storage octets; quantum is
                     an octet --
      CDATA       -- Lextype: (marker,marker)+ --
                  -- Constraint: interpreted as HyTime dimlist --
      "1 -1"      -- Default: entire object --

   zapeof         -- Zap end-of-file --
      (zapeof|nozapeof)
      zapeof

   smcrd          -- Storage manager character reference delimiter --
      CDATA       -- Lextype: char --
      #IMPLIED    -- Default: none --

   compress       -- Compression information --
      CDATA       -- Constraint: registered value notation --
      #IMPLIED    -- Default: none --

   encrypt        -- Encryption information --
      CDATA       -- Constraint: registered value notation --
      #IMPLIED    -- Default: none --
   seal           -- Integrity information --
      CDATA       -- Constraint: registered value notation --
      #IMPLIED    -- Default: none --

   SOIbase        -- Base for relative SOI --
      CDATA       -- Constraint: It is an SOI for this SM --
      #IMPLIED    -- Default: current storage object if it has the
                     same SM, else none --
>
<!notation md5
   PUBLIC "-//IETF/RFC1544//NOTATION FSIDR SEAL
           Content-MD5 Header Field//EN"
>
```

A.6.6 Entity usage attribute definitions

When the entity usage storage manager is declared, an FSI can begin with an entuse start-tag in which attributes can be specified. Attributes appropriate for the entuse tag are those that apply to an entire entity, but are related to its use or storage, rather than its data representation, and are therefore inappropriate to define as data attributes of the entity's notation.

The attribute **modification level** (*modlevel*) allows a record of modification levels to be kept. When a container is used, the modification level for contained objects can be compared to that of the container to ascertain the integrity of the entity. The modification level is a pair of integers. The first integer has an initial value of zero (“0”) and is incremented when the entity is modified. The second integer is set to the value of the first integer when the modified entity is validated by the application.

NOTE 519 For example:

```
<!ENTITY ctr1
   SYSTEM "<entuse modlevel='1 1'>
           <osfile>ctr1.dat"
   NDATA sbento
>
<!ENTITY doc
   SYSTEM "<entuse modlevel='3 3'>
           <sbento in=ctr1>5000"
   NDATA RTF
>
<!ENTITY fig
   SYSTEM "<entuse modlevel='1 1'>
           <sbento in=ctr1 after=doc>2500"
   NDATA CGM
>
<!ENTITY pic
   system "<entuse modlevel='2 2'>
           <sbento in=ctr1 after=fig>2000"
   NDATA JPG
>
```

The entuse tag for the ctr1 sbento container entity indicates that modification level 1 of the container has been validated as correct (the second number matches the first).

The notation parameter of the doc entity indicates it is in Rich Text Format (RTF). The sbento tag states that the entity is stored in the first 5000 octets of ctr1.dat. The entuse tag shows that the entity is at modification level 3, and the second “3” indicates that the entity has been validated since its last modification.

Similarly, the entuse tags for the fig and pic entities identify them as the validated first and second versions, respectively.

The attribute **machine-dependent word information** (*machword*) specifies machine-specific properties of an entity's encoding. Those properties consist of the number of bits in a “byte”, the “word” size used in the encoding, and the permutation of “bytes” in the “word”. The word size could be 2, 4, 8, or more “bytes” (commonly referred to as half-words, words, and doublewords, respectively).

The attribute is used for entities whose notation, though logically portable, is encoded as a sequence of words, with the words being stored in a machine-specific way. It allows the entity manager to transform each word into the form required by the machine that is processing the entity.

NOTE 520 In some cases, these or other properties of an entity, including compression techniques, some aspects of encoding, and some forms of interleaving, may be bound to its notation intrinsically, or to its storage manager. It might not be useful, or even possible, to specify such properties as attributes.

```
                   <!-- Entity Usage Attributes -->
            <!-- THIS IS A NON-MANDATORY STARTER SET. -->
<!NOTATION
   entuse         -- Entity usage start-tag --
                  -- Clause: A.6.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSIDR SYNTAX
           Entity Usage Start-tag//EN"
>
<!ATTLIST #NOTATION
   entuse         -- Entity usage start-tag --
                  -- Clause: A.6.6 --

   included       -- Entities included from notation data --
      CDATA       -- Lextype: ENTITY* --
      #IMPLIED    -- Default: no inclusions --

   modlevel       -- Modification level --
                  -- 1st: 0 when created; incremented when modified.
                     2nd: Set to 1st when validated by application.--
      NUMBERS     -- Lextype: (NUMBER,NUMBER) --
      #IMPLIED    -- Default: not tracked --

   machword       -- Machine-dependent word information --
                  -- AKA "big-endian" or "little-endian" --
                  -- Constraint: byte size and word size, expressed as
                     permutation of normalized word of 2, 4, or 8
                     bytes.  MSB is leftmost and numbered "1" --
                  -- Normalized="8 12", "8 1234", "8 12345678"
                     Example: Intel="8 21", "8 2143", "8 21436587" --
      NUMBERS     -- Lextype: (NUMBER,NUMBER) --
      "8 12"
>
```

A.6.7 Storage manager notation forms

A storage manager is a program (or a combination of programs or a portion of a program) that manages the storage of real physical objects. It works with the entity manager, which manages the mapping of real storage to the virtual storage objects known as entities and acts as an interface between the SGML parser and the storage managers.

Each storage manager name (SMName) specified in an FSI declaration must be declared as a notation name in a notation declaration in the FSI definition document. The declaration identifies the storage manager specification, which defines the syntax and semantics of the storage object specifications for that storage manager. Associated with the notation declaration can be an attribute definition list declaration for “storage manager attributes”.

NOTE 521 It is not necessarily an error if the fsidd cannot be accessed, as an implementation might not require access to it. The primary purpose of the declarations is to advise humans of the storage managers that are used and to declare any attributes that they require. This information is normally built into the relevant software.

In addition to being derived from the base notation forms defined in this architecture, storage manager notation declarations are recognized as such by the use of a formal public identifier in which two uppercase reserved words occur at the start of the public text description component. The words are separated from one another and from the rest of the public text description (after normalization) by a single SPACE. The first word is “FSISM” and the second is one of four storage manager portability class names: LOCAL, PORTABLE, GLOBAL, or CONTAINER. (These are defined in the following sub-clauses.)

A starter set of standardized storage manager declarations and attributes are defined below. Use of them is optional and is not required for conformance to the FSIDR requirements. Moreover, if used, their attribute definition lists can be augmented by any of the attributes defined in *A.6.5 Storage manager attribute definitions*, or other, non-standardized, attributes. However, non-standardized storage managers, attributes, or registered attribute values should not be given the same names as standardized ones.

The default values given for standardized attributes (like those of architectural form attributes) apply only if the attribute is not defined for a storage manager. An fsidd can specify any valid value as the default, or make the attribute required.

NOTE 522 In the declarations of the starter-set storage manager notations, the attribute FSIBase indicates the base notation form defined in this Annex from which a notation form is derived. For example, the storage manager “OSFile” is derived from the local storage manager (localsm) notation form), as indicated by the value “localsm” for its FSIBase attribute.

A.6.7.1 Local storage manager notation form

The notation form **local storage manager** (*localsm*) is used for storage managers that are local to a particular system. An SOS of a local storage manager will normally need to be converted when an FSI in which it occurs (and the storage object it identifies) is moved to another system. For some SMs and operating systems, the needed conversion could be purely syntactic, or even non-existent, if the originating system's directory structure is replicated on the receiving system.

```
                    <!-- Local storage manager -->
<!notation
   localsm        -- Local storage manager --
                  -- Clause: A.6.7.1 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM
           Local Storage Manager//EN"

-- Attributes: smcommon --
>
```

A.6.7.2 Starter set local storage managers

The starter set local storage managers are:

osfile The SOI is an absolute or relative file identifier (that is, possibly including drive, path, filename, extension, etc.) in a form recognized by the local operating system's file open function.

osfd

The SOI is a number. The storage object specification locates the storage object that is created when the operating system reads from the file descriptor with that number. For example, in Unix and DOS systems, fd:0 will read the storage object from standard input.

The content of an osfile SOS (that is, the SOI) conforms to the rules of the local operating system. Where those rules allow an SOI to be relative (e.g. an operating system's relative filename), it normally is interpreted relative to the current storage object when the specified and current storage managers are the same.

```
                   <!-- Local Storage Managers -->
             <!-- PART OF A NON-MANDATORY STARTER SET -->
<!notation
   osfile         -- Operating system file --
                  -- Clause: A.6.7.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM LOCAL
           Operating System File//EN"

-- Attributes: osfile, smcommon --
>
<!attlist #NOTATION
   osfile         -- Operating system file --
                  -- Clause: A.6.7.2 --

   FSIBase  NAME     #FIXED localsm
>

<!notation
   osfd           -- Operating system file desciptor --
                  -- Clause: A.6.7.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM LOCAL
           Operating System File Descriptor//EN"

-- Attributes: osfd, smcommon --
>
<!attlist #NOTATION
   osfd           -- Operating system file desciptor --
                  -- Clause: A.6.7.2 --

   FSIBase  NAME     #FIXED localsm
>
```

A.6.7.3 Portable storage manager notation form

The notation form **portable storage manager** (*portblsm*) is used for storage managers whose SOSs are recognized on many (but not necessarily all) operating systems. A portable SOS will normally not need to be converted when an FSI containing it is moved to another system.

```
                  <!-- Portable Storage Manager -->
<!notation
   portblsm       -- Portable storage manager --
                  -- Clause: A.6.7.3 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM
```

```
           Portable Storage Manager//EN"

-- Attributes: smcommon --
>
```

A.6.7.3.1 URL Portable storage manager

The portable storage manager “URL” uses SOSs that are uniform resource locators conforming to Internet RFC 1738.

```
                  <!-- Portable Storage Managers -->
            <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   URL            -- Uniform Resource Locatior --
                  -- Clause: A.6.7.4 --

   PUBLIC "-//IETF/RFC1738//NOTATION FSISM PORTABLE
           Uniform Resource Locator//EN"

-- Attributes: URL, smcommon --
>
<!attlist #NOTATION
   URL            -- Uniform Resource Locator --
                  -- Clause: A.6.7.4 --

   FSIBase  NAME     #FIXED portblsm
>
```

A.6.7.3.2 Neutral file identifier storage manager

The notation **neutral storage manager** (*neutral*) is a storage manager for which the SOSs can be reliably transformed into system-specific (local) SOSs). The neutral file identifier storage manager operates by transforming its SOS to the format used by the SM of the current storage object, which then performs the actual storage access. If that SM is a container SM, its “in” attribute is used in the transformed SOS. All other transformed SOS attributes come from the neutral SOS.

The transformation treats the neutral SOI as a hierarchical name whose components are separated by slashes. (There is no initial slash.)

NOTE 523 For example, for each of the storage managers in the following list, the SOS “<neutral>entities/e1.sgm” would be transformed as shown:

osfile on a DOS system
<osfile>entities\e1.sgm

osfile on a Mac system
<osfile>:entities:e1.sgm

osfile on a Unix system
<osfile>entities/e1.sgm

url <url>entities/e1.sgm

The attribute **fold** (*fold*) can be specified to show that the neutral SOI is to be “folded” (that is, its case transformed) to comply with the case conventions of the current SM.

NOTE 524 For example, on a Unix system, filenames are case-sensitive but normal files are conventionally named in all lowercase. Therefore, the SOS “<neutral fold>FOO.SGM” would be transformed to “<osfile>foo.sgm” whereas “<neutral nofold>FOO.SGM” would be transformed to “<osfile>FOO.SGM”.

NOTE 525 Not every neutral SOI will be transformable to a given SM format. A neutral SOI is most likely to be transformable when fold is specified and each of its components is an alphanumeric filename, or a filename and alphanumeric extension separated by a period, and the length of a filename is limited to eight characters and that of an extension to three characters.

```
                   <!-- Neutral File Identifier -->
            <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   neutral        -- Neutral file identifier --
                  -- Clause: A.6.7.4.1 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM PORTABLE
           Neutral File Identifier//EN"

-- Attributes: neutral, smcommon --
>
<!attlist #NOTATION
   neutral        -- Neutral file identifier --
                  -- Clause: A.6.7.4.1 --

   FSIBase  NAME     #FIXED portblsm

   fold           -- Fold neutral SOI to current SM conventions? --
                  -- Constraint: direction of folding is dependent on
                     semantics of target storage manager SOS is
                     transformed to. --
      (fold|nofold)
      fold
>
```

A.6.7.3.3 Notation processor storage manager notation form

The notation form **notation processor storage manager** (*dcnsm*) is used for storage managers that manage the association of data content notations with processors. An SOS of notation storage manager class can be used only in a system identifier that is part of the external identifier for a notation. An SOS of this form identifies a notation to a system in a way that allows the system to process data that is in that notation.

```
             <!-- Notation Processor Storage Manager -->
            <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   dcnsm          -- Notation processor storage manager --
                  -- Clause: A.6.7.3.3 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM PORTABLE
           Notation Storage Manager//EN"

-- Attributes: dcnsm, smcommon --
>
<!attlist #NOTATION
   dcnsm          -- Notation processor storage manager --
```

```
                  -- Clause: A.6.7.3.3 --

   FSIBase  NAME     #FIXED portblsm
>
```

A.6.7.3.4 Notation processor storage managers

The starter set notation storage manager is:

mimetype   The SOI is the MIME type and subtype optionally followed by parameters, specified as they would be in the field body of a Content-Type header field in the header of an Internet mail message.

```
        <!-- MIME Type Notation Processor Storage Managers -->
            <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   mimetype       -- MIME content type --
                  -- Clause: A.6.7.4.3 --

   PUBLIC "-//IETF/RFC1521//NOTATION FSISM PORTABLE
           MIME Content Type//EN"

-- Attributes: mimetype, smcommon --
>
<!attlist #NOTATION
   mimetype       -- MIME content type --
                  -- Clause: A.6.7.4.3 --

   FSIBase  NAME     #FIXED dcnsm
>
```

A.6.7.4 Global storage manager notation form

The notation form **global storage manager** (*globalsm*) is used for storage managers whose SOSs never need to be converted because they are independent of any real storage environment.

```
                   <!-- Global Storage Manager -->
<!notation
   globalsm       -- Global storage manager --
                  -- Clause: A.6.7.5 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM
           Global Storage Manager//EN"

-- Attributes: smcommon --
>
```

A.6.7.5 Global storage managers

The starter set global storage managers are:

literal The SOI is treated as the literal text of a storage object.

NOTE 526 Literal text is used chiefly as a connector when concatenating other storage objects. Numeric character references can be used to insert record boundaries between the concatenated objects.

ThisOne The SOI must be empty. The storage object is normally the current storage object. However, if the attribute **this entity** (*entity*) is specified, the “storage object” is the sequence of characters of the entity named in the attribute value, or the current entity if the attribute value is empty.

NOTE 527 ThisOne can be used to make system identifiers portable. In the following example, the entity “ent1” occupies the last 500 octets of the current storage object. If the storage location of that object were changed, it would not be necessary to change the ent1 entity declaration.

```
<!ENTITY ent1 system "<ThisOne extents='-500 -1'>">
```

```
                 <!-- Global Storage Managers -->
          <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   literal        -- Literal storage manager --
                  -- Clause: A.6.7.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM GLOBAL
           Literal Text//EN"

-- Attributes: literal, smcommon --
>
<!attlist #NOTATION
   literal        -- Literal storage manager --
                  -- Clause: A.6.7.6 --

   FSIBase  NAME     #FIXED globalsm
>

<!notation
   ThisOne        -- This storage object storage manager --
                  -- Clause: A.6.7.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM GLOBAL
           This Storage Object//EN"

-- Attributes: ThisOne, smcommon --
>
<!attlist #NOTATION
   ThisOne        -- This storage object storage manager --
                  -- Clause: A.6.7.6 --

   FSIBase  NAME     #FIXED globalsm

   entity         -- Entity whose characters are used --
      CDATA       -- Lextype: ENTITY --
                  -- Constraint: empty string means this entity --
      #IMPLIED    -- Default: the octets of this storage object --
>
```

A.6.7.6 Container storage manager notation form

The notation form **container storage manager** (*contnrsm*) is used for storage managers that contain other entities. Whether a container storage manager SOS needs to be converted depends on whether it contains system-dependent information such as file identifiers, as is the case for TAR. For sbento, on the other hand, the SOIs are relative to the entity and the SM is therefore effectively global.

NOTE 528 The storage of the container entity itself depends on the SMs used for that purpose, and is independent of the universality of the container notation.

To locate an object in a container, it is necessary to specify both the SOS of the container and of the object within it. As the container could itself be stored in a container, several levels of SOS might be required. While storage manager notations could be defined to provide for such situations, formal system identifiers provide a simpler and more flexible approach by using separate entity declarations for the container and the contained objects.

First, the container is declared as an entity with a notation that identifies the type of container (that is, the container storage manager). Examples might be “TAR” (Posix Tape ARchive) or “sbento” (see *A.6.7.6.2 Standard BENTO (sbento)*). The SOS of the container entity is in the format of the storage manager in which the container itself is stored.

Second, each entity stored in the container has its own declaration, in which the storage manager name in the SOS tag is the container manager. The SOS tag exhibits the attribute “in”, which identifies the container entity in which the declared entity is stored. The SOI locates the declared entity in the container. In a typical archive, the SOI might be a file identifier.

NOTE 529 Container entities can be nested. A subordinate (inner) container entity can either have the same container notation as the outer container or a different one.

NOTE 530 A query language (such as SQL or SDQL) can be used as a container notation. In such cases, only a query that returns a sequence of characters is valid as an SOI.

The attribute **in container** (*in*) identifies the container entity in which the specified storage object (the contained object) occurs. It can be specified as either an entity name or a formal system identifier (although the latter form should be used only where an entity name is not available, such as on a command line). Entities named on “in” attributes must be declared before the declarations of entities that refer to them.

The attribute **previous member of container** (*after*) names the member of the container that the current entity comes after. If the after attribute is not specified, the position of the entity in the container must be sufficiently specified in the entity's SOS or the entity is the first member of the container if no position is specified in the SOS.

NOTE 531 The after attribute is intended primarily for container storage managers that identify contained objects by relative or absolute position rather than by name. However, some storage managers may choose to allow access by name and relative position.

```
                 <!-- Container Storage Managers -->
<!notation
   contnrsm       -- Container storage manager --
                  -- Clause: A.6.7.7 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM GLOBAL
           Container Storage Manager//EN"

-- Attributes: after, contnrsm, in --
-- CommonAttributes: smcommon --
>
<!attlist #NOTATION
   contnrsm       -- Container storage manager --
```

```
                  -- Clause: A.6.7.7 --

   FSIBase  NAME     #FIXED globalsm
>
<!attlist #NOTATION
-- in --           -- In container --
                  -- Clause: A.6.7.7 --
   (contnrsm,tar,mime,sbento)

   in             -- In container --
                  -- Container in which this object is stored --
                  -- Constraint: declaration for container entity
                     named must precede declaration of contained
                     entity. --
      CDATA       -- Lextype: (ENTITY|fsi) --
      #REQUIRED
>
<!attlist #NOTATION
-- after --       -- Previous member of container --
                  -- Clause: A.6.7.7 --
   (contnrsm,sbento)

   after          -- Previous member of container --
                  -- Member of container this entity comes after
                     (follows) --
      CDATA       -- Lextype: (ENTITY|fsi) --
      #IMPLIED    -- Default: position sufficiently specified by SOS
                     or entity is first member of container. --
>
```

A.6.7.6.1 Container storage managers

The starter set of container storage managers is:

tar	Posix tape archive.
mime	Mime single-part or multi-part message.
sbento	Standard bento. Described under *A.6.7.6.2 Standard BENTO (sbento)*.

```
                 <!-- Container Storage Managers -->
            <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   tar            -- Posix Tape Archive --
                  -- Clause: A.6.7.7.1 --
                  -- Creates container by partitioning storage object
                     to emulate file system directory structure. SOI
                     is a file identifier. --

   PUBLIC "ISO/IEC 9945-1:1990//NOTATION FSISM CONTAINER
           Posix Tape Archive//EN"

-- Attributes: tar, in --
-- CommonAttributes: smcommon --
>
```

```
<!attlist #NOTATION
   tar            -- Posix Tape Archive --
                  -- Clause: A.6.7.7.1 --

   FSIBase  NAME     #FIXED contnrsm
>

<!notation
   MIME           -- MIME storage manager --
                  -- Clause: A.6.7.7.1 --
                  -- Creates container by separating stored entities
                     with a unique boundary string, identifying them
                     with headers, and possibly applying a content
                     transfer encoding. SOI is a MIME content-ID for a
                     multi-part message and is empty for a single-part
                     message. --

   PUBLIC "-//IETF/RFC1521//NOTATION FSISM CONTAINER
           Multimedia Internet Mail Extensions//EN"

-- Attributes: MIME, in --
-- CommonAttributes: smcommon --
>
<!attlist #NOTATION
   MIME           -- MIME storage manager --
                  -- Clause: A.6.7.7.1 --

   FSIBase  NAME     #FIXED contnrsm
>
```

A.6.7.6.2 Standard BENTO (sbento)

Sbento is a container storage manager with the property that it does not utilize the names of its contained entities to store them. The SOI locates them solely by their position and size in the container entity, using a dimension specification list that specifies the extents of the container occupied by the contained entity. Multiple dimension specifications can be specified if the contained entity is segmented and distributed in several locations within the container. The notation for the dimension specification list is the HyTime marker list notation (*marklist*) (see *6.8 Coordinate Specifications*).

NOTE 532 The name “sbento” comes from the Japanese word “bento”: “A box or basket with multiple compartments, containing a collection of disparate entities arranged in an esthetically pleasing manner.” It is an acronym for “Standard Bento Entity for Natural Transport of Objects”.

NOTE 533 Sbento does not prohibit overlapping of contained objects; it is for the application to determine whether overlapping is valid. Overlapping can be a useful technique, for example, for identifying the color table in a graphics entity. The color table would be declared as a separate entity, but its offset and size would position it within the storage occupied by the graphics entity.

NOTE 534 In the following example, the sbento container entity “ctr1” is stored in the local operating system file “ctr1.dat”. The entity “doc” occupies the first 5000 characters in “ctr1”, while the entity “fig” occupies the following 2500 characters.

```
<!ENTITY ctr1
   SYSTEM "<osfile>ctr1.dat"
   NDATA sbento
>
<!ENTITY doc
   SYSTEM "<sbento in=ctr1>1 5000"
>
```

```
<!ENTITY fig
   SYSTEM "<sbento in=ctr1>5001 2500"
>
```

If the dimension specification in the SOI is a single number, the first is deemed to be omitted and is calculated to be one greater than the position of the last character of the entity named by the after attribute.

NOTE 535 This rule follows the normal marker list notation rule for an omitted first marker. It allows the following variation on the previous example:

```
<!ENTITY ctr1
   SYSTEM "<osfile>ctr1.dat"
   NDATA sbento
>
<!ENTITY doc
   SYSTEM "<sbento in=ctr1>5000"
>
<!ENTITY fig
   SYSTEM "<sbento in=ctr1 after=doc>2500"
>
```

When the sbento container is the document entity in which the sbento container is declared, the common attribute *extents* is used to specify the actual storage octets of the document entity in which the sbento container entity occurs that make up the actual sbento container data (see *A.6.5.3 Common storage manager attributes*). Normally, the second marker is specified as “-1”, indicating that the container occupies all the remaining octets of the storage object. If the sbento data is not contiguous, multiple axis marker pairs may be specified.

NOTE 536 For example, an sbento container document might have a doctype declaration and document element start-tag that together occupy the first 341 octets of the document entity's storage object. The sbento container would start at octet 342 and occupy the remaining octets. Thus the sbento container entity would be declared as:

```
<!ENTITY bento1
   SYSTEM "<ThisOne extents='342 -1'>"
   NDATA sbento
>
```

```
   <!-- Standard Bento Entity for Natural Transport of Objects -->
            <!-- PART OF A NON-MANDATORY STARTER SET. -->
<!notation
   sbento         -- Standard BENTO storage manager --
                  -- Clause: A.6.7.7.2 --
                  -- Creates container by partitioning storage object
                     into characters that can be occupied by
                     entities. SOI is number list that counts
                     characters.
                     SOI constraint: interpreted as HyTime dimlist.
                     SOI lextype: (marker|(marker,marker)+) --

   PUBLIC "ISO/IEC 10744:1997//NOTATION FSISM CONTAINER
          Standard BENTO//EN"

-- Attributes: after, in, sbento --
-- CommonAttributes: smcommon --
>
<!attlist #NOTATION
   sbento         -- Standard BENTO storage manager --
                  -- Clause: A.6.7.7.2 --
   FSIBase  NAME     #FIXED contnrsm
>

       <!-- TEMPLATE FOR DECLARING SBENTO CONTAINER ENTITY -->
<!-- Container shares storage object with SGML document entity in whose
     DOCTYPE internal subset the sbento entity declarations occur. The
     extents value "nnn" in the template is replaced by the position of
     the first storage octet after the SGML document entity. -->

<!ENTITY bento1
   SYSTEM "<ThisOne extents='nnn -1'>"
   NDATA sbento
>

       <!-- TEMPLATE FOR DECLARING SBENTO CONTAINED ENTITY -->
<!-- Contained entity occupies first 500 units of "bento1" entity. -->
<!ENTITY inbento1
   SYSTEM "<sbento in=bento1>1 500"
>
```

A.6.8 Conformance

A document or system can support the use of Formal System Identifiers in accordance with the requirements of this annex without conforming to other requirements of this International Standard.

The storage manager notations and attributes standardized in this annex comprise a non-mandatory starter set. Their use is not required for conformance. Other notations and/or attributes can be used in addition to, or instead of, the standardized ones.

Allowable tokens for the value of the FSI support attribute fsidr are:

```
fsism="altsos smalias validate"
```

where:

altsos	An FSI can include alternative SOS sequences.
smalias	FSIs in the document can use alias names for the SMs declared in the FSI definition document.
validate	The system reports an error if an FSIDR requirement is not met.

A system that conforms to these FSIDR requirements shall so indicate by including the following statement in its system FSI definition document:

```
A Formal System Identifier implementation
conforming to the FSIDR requirements of
International Standard ISO/IEC 10744.
```

Support for entuse, sbento, and other standardized storage manager notations is indicated by including their declarations in the system FSI definition document. The fsidd is considered part of the system's conforming documentation.

A.7 SGML Property Set

The SGML property set defined in this clause includes every class and property recognizable by an SGML parser, both in the abstract document described by SGML markup and in the markup itself.

The syntax and semantics of property definition documents are described in *A.4 Property Set Definition Requirements (PSDR)*.

A.7.1 SGML Notation

The notation SGML is used to declare SGML documents addressed from another SGML document.

The data attribute **active DTDs or LPDs** (*active*) names the document types and/or link types to be used when parsing the document entity.

NOTE 537 When more than one active document type is specified, the document is parsed against each document type and separate grove for each parse is produced. When multiple LPDs and DTDs are specified, the entire list of active LPDs is applied to the parse for each DTD.

The data attribute **active architecture** (*arch*) names the architecture to be used when processing the document. When there is a hierarchy of architectures from which a client document is derived, a specific architecture may be specified by naming the architectures by architecture notation name, starting with a base architecture of the client document. The effect of addressing a document entity processed according to a specific architecture is to address the architectural instance grove derived from the document, rather than the document itself.

NOTE 538 In other words, the arch attribute enables explicit addressing of architectural document instances.

```
                    <!-- SGML Data Attributes -->
<!notation
   SGML           -- Standard Generalized Markup Language --
                  -- Clause: A.5.7.2 --
                  -- For SGML entities declared as CDATA --

   PUBLIC "ISO 8879:1986//NOTATION
           Standard Generalized Markup Language//EN"

-- Attributes: SGML --
-- CommonAttributes: altreps, included, superdcn --
>
<!attlist #NOTATION
   SGML           -- Standard Generalized Markup Language --
                  -- Clause: A.5.7.2 --

   active         -- Active DTDs or LPDs for SGML document/subdoc. --
      CDATA       -- Constraint: DTD or LPD names --
      #IMPLIED    -- Default: base DTD --

   arch           -- Active architecture for document/subdoc --
      CDATA       -- Constraint: name of architecture of which
                     document is directly or indirectly a client --
      #IMPLIED    -- Default: no active architecture --
>
```

A.7.2 SGML property set

The SGML property set is defined in the following document. This document is identified by the public identifier “ISO/IEC 10744:1997//DOCUMENT SGML Property Set//EN”.

```
<!-- SGML Property Set -->

<!DOCTYPE propset
   PUBLIC "ISO/IEC 10744:1997//DTD Property Set//EN"
[
<!NOTATION SGML
   PUBLIC "ISO 8879:1986//NOTATION
           Standard Generalized Markup Language (SGML)//EN"
>
<!NOTATION SGMLGC
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           SGML Grove Construction Process//EN"
>
<!NOTATION DSSSL
   PUBLIC "ISO/IEC 10179:1996//NOTATION
           Document Style Semantics and Specification Language
           (DSSSL)//EN"
>
<!NOTATION HyTime
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
]>

<propset nsd=SGML gcsd=SGMLGC>
<desc>
Defines the classes and properties to be used in the construction of
groves from the parsing of SGML documents.

Classes and properties are classified as follows:
  o Abstract or SGML document string (SDS)
  o SGML declaration, document prolog, or document instance
  o Required only for support of datatag, rank, shortref, link, subdoc,
    fpi, architectures, notset, fsi, genarc, pelement.

ESIS corresponds roughly to the combination of baseabs (base abstract),
prlgabs0, and instabs (instance abstract).
</desc>

<!-- String Normalization Rules -->

<normdef rcsnm=general clause="d4506">
<desc>
Declared concrete syntax general namecase substitution.

<normdef rcsnm=entity clause="d4506">
<desc>
Declared concrete syntax entity namecase substitution.

<normdef rcsnm=rcsgener clause="d4506">
<desc>
```

```
Reference concrete syntax general namecase substitution.

<!-- Base abstract classes and properties -->

<psmodule rcsnm=baseabs fullnm="base abstract" default>

<classdef rcsnm=sgmldoc appnm="sgml document" clause="62001">
<desc>
The parsed SGML document or subdocument.  The root of the grove.

<propdef subnode rcsnm=sgmlcsts appnm="sgml constants" node
ac=sgmlcsts clause="41170 41180">

<propdef rcsnm=appinfo appnm="application info"
fullnm="application information" string
clause="d6001">
<desc>
Application information provided by the SGML declaration.
<when>
A literal was specified as the value of the APPINFO parameter
of the SGML declaration applicable to the document/subdocument.

<propdef subnode rcsnm=prolog nodelist
ac="doctpdcl lktpdcl comdcl pi ssep" clause="71001">

<propdef subnode rcsnm=epilog nodelist ac="comdcl pi ssep"
clause="71002">
<desc>
Other prolog following the document instance.

<classdef rcsnm=sgmlcsts appnm="sgml constants" clause="b6004 c2101">
<desc>
A holding pen for selected nodes intrinsic to all SGML documents,
which may be needed as irefnodes elsewhere.
<note>
This has no properties unless the srabs (shortref abstract)
or linkabs (link abstract) modules are included.

<classdef rcsnm=attasgn appnm="attribute assignment"
conprop=value dsepprop=tokensep clause="79002">
<desc>
An attribute assignment, whether specified or defaulted.
<note>
In the base module because of data attributes.

<propdef subnode rcsnm=value nodelist
ac="attvaltk datachar sdata intignch entstart entend" clause="79401">
<note>
If the attribute value is tokenized, the children are of type attvaltk;
otherwise, they are of the other allowed types.
<when>
The attribute is not an impliable attribute for which there is no
attribute specification.

<propdef rcsnm=name string strnorm=general
clause="93001">
```

```
<propdef rcsnm=implied boolean clause="b3407">
<desc>
True if and only if the attribute is an impliable attribute
for which there is no attribute specification.

<propdef rcsnm=tokensep appnm="token sep" fullnm="token separator"
char clause="79400">
<desc>
The separator between the tokens of the value. Always equal
to the SPACE character in the concrete syntax.
<when>
The node has two or more children of class attvaltk.

<classdef rcsnm=attvaltk appnm="attribute value token" conprop=token
clause="79305">

<propdef rcsnm=token string clause="93003">

<classdef rcsnm=datachar appnm="data char" fullnm="data character"
conprop=char clause="92002">

<propdef rcsnm=char fullnm=character char clause="92003">
<desc>
The character returned by the parser to the application.
<when>
The character is not the result of a numeric character reference to a
non-SGML character.

<propdef rcsnm=nonsgml appnm="non sgml bit combination" integer>
<when>
The character is the result of a numeric character reference to a non-SGML
character.

<classdef rcsnm=sdata
fullnm="internal specific character data entity reference result"
conprop=char clause="92101">

<propdef rcsnm=sysdata appnm="system data" string clause="43041">
<note>
The replacement text of a specific character data entity is treated
as system data when referenced.

<propdef rcsnm=char fullnm=character char sd=DSSSL>
<desc>
The character associated with the SDATA entity by the map-sdata-entity
architectural form.
<when>
A character has been associated with the SDATA entity by the
map-sdata-entity architectural form.

<classdef rcsnm=pi fullnm="processing instruction" clause="80000">
<desc>
Processing instruction.

<propdef rcsnm=sysdata appnm="system data" string clause="80002">
```

```
</psmodule>

<!-- Prolog-related abstract classes and properties, level 0 -->

<psmodule rcsnm=prlgabs0 fullnm="prolog abstract level 0" default
dependon=baseabs>

<propdef irefnode rcsnm=govdt appnm="governing doctype" node
ac=doctype
cn=sgmldoc clause="71004">
<desc>
The document type that governs the parse.  When there are more than one
"active" document types specified, each active document type gives rise
to a separate parse, which, in turn, creates a separate sgmldoc grove.

<propdef subnode rcsnm=dtlts appnm="doctypes and linktypes"
fullnm="document types and link types"
nmndlist ac="doctype linktype" acnmprop="name name" cn=sgmldoc
clause="71001">
<desc>
The document types and link types declared in the prolog, in declaration
order.

<classdef rcsnm=doctype appnm="document type" clause="b1000">
<desc>
The abstraction of a document type declaration.
<note>
It includes entities declared in that declaration's DTD,
entities treated as being declared therein because they
occur in a link type for which that DTD is the source DTD,
and entities declared in the base declaration which may be
referenced when this document type is active.

<propdef rcsnm=name string strnorm=general clause="b1002">
<desc>
The name associated with the DTD by the document type declaration;
necessarily also the name of the type of the outermost element.

<propdef rcsnm=govrning appnm=governing boolean clause="71005">
<desc>
True if either this was the active document type or there was
no active document type and this is the base document type.
<note>
The "governing" document type governs the parsing process.
If more than one document type is specified as "active",
each active document type gives rise to a separate parse,
for which it is the governing document type, and thereby
produces a separate grove.

<propdef subnode rcsnm=genents appnm="general entities" nmndlist
ac=entity acnmprop=name clause="b1004">
<desc>
The general entities of the document or subdocument declared in the DTD.
<note>
Includes entities not explicitly declared, as discussed above
```

```
in the description of this class.
<note>
If the DTD provides a default declaration for undeclared general
entity names, there is no entry in the list corresponding to this
declaration, nor any entry for any such undeclared name, except for
those defaulted entities referenced within the DTD (e.g. in CDATA
attribute values).  Other defaulted entities are in the entities
property of the sgmldoc class.  See dfltent following.

<propdef subnode rcsnm=nots appnm=notations nmndlist ac=notation
acnmprop=name clause="b1005">

<classdef rcsnm=entity clause="60000">

<propdef rcsnm=name string strnorm=entity clause="93001">

<propdef rcsnm=enttype appnm="entity type" enum clause="a5502">

<enumdef rcsnm=text fullnm="SGML text">
<enumdef rcsnm=cdata>
<enumdef rcsnm=sdata>
<enumdef rcsnm=ndata>
<enumdef rcsnm=subdoc appnm=subdocument>
<enumdef rcsnm=pi>

<propdef rcsnm=text fullnm="replacement text" string clause="92101">
<when>
The entity is an internal entity.

<propdef subnode rcsnm=extid appnm="external id" fullnm="external identifier"
node ac=extid clause="a1601">
<when>
The entity is an external entity.

<propdef subnode rcsnm=atts appnm=attributes
nmndlist ac=attasgn acnmprop=name clause="b4120">
<desc>
A list of data attribute assignments, one for each declared attribute of
the entity in the order in which they were declared in the attribute
definition list declaration.
<when>
The entity is an external data entity.

<propdef rcsnm=notname appnm="notation name" string
strnorm=general clause="79408">
<when>
The entity is an external data entity.

<propdef irefnode rcsnm=notation node ac=notation clause="b4001">
<when>
The entity is an external data entity.

<classdef rcsnm=notation fullnm="data content notation" clause="b4000">

<propdef rcsnm=name string strnorm=general clause="79441">
```

```
<propdef subnode rcsnm=extid appnm="external id" fullnm="external identifier"
node ac=extid clause="a1601">

<classdef rcsnm=extid appnm="external id" fullnm="external identifier"
clause="a1600">

<propdef rcsnm=pubid appnm="public id" fullnm="public identifier"
string  clause="a1602">
<when>
The external identifier contained an explicit public identifier.

<propdef rcsnm=sysid appnm="system id" fullnm="system identifier"
string clause="a1603">
<when>
The external identifier contained an explicit system identifier.

<propdef optional rcsnm=gensysid appnm="generated system id"
fullnm="generated system identifier"
string>
<desc>
The system identifier generated by the system from the external
identifier and other information available to the system.
<when>
The external identifier is not the external identifier of
the default entity.

</psmodule>

<!-- Document instance related abstract classes and properties -->

<psmodule rcsnm=instabs fullnm="instance abstract" default
dependon=baseabs>

<propdef subnode rcsnm=docelem appnm="document element" node
ac=element cn=sgmldoc clause="72003">
<desc>
The document element for the governing document type.

<propdef irefnode rcsnm=elements nmndlist ac=element acnmprop=id
cn=sgmldoc clause="73001">
<desc>
All the elements in the document which have unique identifiers in the
order in which they are detected by the parser: parents occur
before children; siblings occur in left-to-right order.

<propdef irefnode rcsnm=entities nmndlist ac=entity acnmprop=name
cn=sgmldoc clause="94410">
<desc>
The explicitly declared general entities from the governing document
type, followed by the defaulted entities.
<note>
This includes both internal and external entities. It does not
include unnamed entities.

<propdef subnode rcsnm=dfltents appnm="defaulted entities" nmndlist
ac=entity acnmprop=name cn=sgmldoc clause="94412">
```

```
<desc>
An entity for each entity name in the document that referenced
the default entity in the governing document type.

<!-- Attribute value token -->

<propdef irefnode rcsnm=entity node ac=entity cn=attvaltk
clause="79401">
<when>
Declared value of attribute is ENTITY or ENTITIES.

<propdef irefnode rcsnm=notation node ac=notation cn=attvaltk
clause="79408">
<when>
Declared value of attribute is NOTATION.

<propdef irefnode rcsnm=referent node ac=element cn=attvaltk
clause="79403">
<when>
Declared value is IDREF or IDREFS.

<classdef rcsnm=element conprop=content clause="73000">

<propdef rcsnm=gi fullnm="generic identifier" string
strnorm=general clause="78001">
<desc>
Generic identifier (element type name) of element.

<propdef derived rcsnm=id fullnm="unique identifier" string
 strnorm=general clause="79403">
<when>
A unique identifier was specified for the element.

<propdef subnode rcsnm=atts appnm=attributes
nmndlist ac=attasgn acnmprop=name clause="79301">
<desc>
A list of attribute assignments, one for each declared attribute
of the element in the order in which they were declared in the
attribute definition list declaration.

<propdef subnode rcsnm=content nodelist
ac="datachar sdata element pelement extdata subdoc pi msignch ignrs
    ignre repos usemap uselink entstart entend ssep comdcl msstart
    msend ignmrkup"
clause="76001">

<classdef rcsnm=extdata appnm="external data"
fullnm="reference to external data" clause="a5500">
<desc>
The result of referencing an external data entity.

<propdef rcsnm=entname appnm="entity name" string
strnorm=entity clause="a5101">

<propdef irefnode rcsnm=entity node ac=entity clause="94410">
```

```
</psmodule>

<!-- Base SDS classes and properties -->

<psmodule rcsnm=basesds0 fullnm="base SGML document string level 0"
dependon=baseabs>

<!-- Sdata -->

<propdef optional rcsnm=entname appnm="entity name" string
 strnorm=entity cn=sdata clause="a5101">

<propdef irefnode rcsnm=entity node ac=entity cn=sdata
clause="94410">

<!-- Processing instruction -->

<propdef rcsnm=entname appnm="entity name" string
strnorm=entity cn=pi clause="a5101">
<when>
The processing instruction resulted from referencing a PI entity.

<propdef irefnode rcsnm=entity node ac=entity cn=pi
clause="94410">
<when>
The processing instruction resulted from referencing a PI entity.

<!-- Entity -->

<propdef rcsnm=dflted appnm=defaulted boolean cn=entity
clause="94412">
<desc>
True if this was created because of a reference to the default entity.

</psmodule>

<psmodule rcsnm=basesds1 fullnm="base SGML document string level 1"
dependon=basesds0>

<propdef subnode optional rcsnm=entref appnm="entity ref"
fullnm="entity reference" nodelist
ac="gendelm name ssep entstart entend refendre shortref" cn=pi
clause="94401">
<desc>
The markup of the entity reference.
<note>
ssep, entstart, and entend may occur only in a name group in a named
entity reference.
<when>
The processing instruction resulted from referencing a PI entity with
a named entity reference or a short reference.

<propdef subnode optional rcsnm=open appnm="open delim"
fullnm="open delimiter" node ac=gendelm cn=pi clause="80001">
<when>
The processing instruction did not result from referencing a PI entity.
```

```
<propdef subnode optional rcsnm=close appnm="close delim"
fullnm="close delimiter" node ac=gendelm cn=pi clause="80001">
<when>
The processing instruction did not result from referencing a PI entity.

<!-- Attribute -->

<propdef irefnode rcsnm=attspec appnm="attribute spec" fullnm="attribute
specification"
nodelist ac="name ssep gendelm literal attvalue" cn=attasgn
clause="79002">
<when>
The attribute was specified rather than defaulted or implied.

<propdef irefnode rcsnm=attvalsp appnm="attribute value spec"
fullnm="attribute value specification" node
ac="attvalue literal" cn=attasgn clause="79301">
<when>
The attribute is not implied.

<!-- Data character -->

<propdef rcsnm=intrplch appnm="interp replaced char"
fullnm="interpretation replaced character" char cn=datachar
clause="a1704">
<desc>
The character that was replaced.
<note>
When a sequence of RE and/or SPACE characters in a minimum literal
is replaced by a single SPACE character, then the first
character is represented by a datachar possibly with an intrplch
property, and the other characters are represented by an intignch.
<when>
The data character replaced another character
when a literal was interpreted: a SPACE character that replaced a
RE or SEPCHAR in an attribute value literal or an RE in a minimum
literal.

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" cn=datachar clause="95001">
<when>
The data character was the replacement of a named character reference.

<propdef subnode optional rcsnm=numcref appnm="numeric char ref"
fullnm="numeric character reference" nodelist
ac="gendelm name crefcnum refendre" cn=datachar clause="95001">
<when>
The data character was the replacement of a numeric character reference.

<!-- Specific character data -->

<propdef subnode optional rcsnm=markup nodelist
ac="gendelm name ssep entstart entend refendre shortref" cn=sdata
clause="94401">
```

```
<note>
ssep, entstart, and entend can occur only in a name group in a named
entity reference.

<classdef rcsnm=ssep appnm="s sep" fullnm="s separator" mayadd
clause="62100">

<propdef rcsnm=char fullnm=character char clause="92003">

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" clause="95001">
<when>
The character was the replacement of a named character reference.

<classdef rcsnm=comment clause="a3002">

<propdef subnode optional rcsnm=open appnm="open delim"
fullnm="open delimiter" node ac=gendelm clause="a3002">

<propdef rcsnm=chars fullnm=characters string clause="92101">
<desc>
The characters in the comment (excluding the com delimiters).

<propdef subnode optional rcsnm=close appnm="close delim"
fullnm="close delimiter" node ac=gendelm clause="a3002">

<classdef rcsnm=comdcl appnm="comment decl" fullnm="comment declaration"
conprop=markup mayadd clause="a3001">

<propdef subnode rcsnm=markup nodelist ac="comment ssep"
clause="a3001">

<classdef rcsnm=ignmrkup appnm="ignored markup" conprop=markup
clause="77002 94405 c3007">
<desc>
Ignored markup.  Either a start-tag or end-tag that is ignored because
it contains a document type specification that contains a name group
none of the names in which is the name of an active document type, or
a general or parameter entity reference that is ignored because it
contains a name group none of the names in which is the name of an
active document or link type, or a link set use declaration that is
ignored because its link type name is not an active link type,
or a general entity reference in an attribute value literal in
a start-tag that is itself ignored markup, or an entity declaration
that is ignored because the entity was already declared.

<propdef subnode rcsnm=markup nodelist
ac="gendelm name ssep attvalue literal entstart entend refendre"
clause="74001 75001 94401 c3001">

<classdef rcsnm=entstart appnm="entity start" conprop=markup>
<desc>
The start of the replacement text of an entity.
<note>
The end shall be marked by an entend node. This is the result of an
```

```
entity reference that was replaced by the parser.

<propdef subnode optional rcsnm=markup nodelist
ac="gendelm name ssep entstart entend refendre shortref">
<desc>
The markup of the entity reference.

<propdef optional rcsnm=entname appnm="entity name" string
 strnorm=entity>

<propdef irefnode rcsnm=entity node ac=entity clause="a5201">

<classdef rcsnm=entend appnm="entity end" clause="94500">
<desc>
The end of an entity reference that was replaced by the parser.

<classdef rcsnm=msignch appnm="marked section ignored char"
fullnm="marked section ignored character" clause="a4204">
<desc>
A character that has been ignored within a marked section.

<propdef rcsnm=char fullnm=character char clause="92101">

<classdef rcsnm=intignch appnm="interp ignored char"
fullnm="interpretation ignored char" clause="79303 a1704">
<desc>
A character in a literal that was ignored when the literal was
interpreted: an RS in an attribute value literal or in a minimum literal,
an RE or SPACE character in a minimum literal that immediately
followed another RE or SPACE character in a minimum literal,
or an RE or SPACE character that was the first or last character
in a minimum literal.

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" clause="95001">
<when>
The character was the replacement of a named character reference.

<propdef rcsnm=char fullnm=character char clause="92101">

<classdef rcsnm=gendelm appnm="general delim" fullnm="general delimiter"
clause="FIG30">
<desc>
A general delimiter.

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" clause="95001">
<note>
This may happen only for a delimiter that is the first child
of its parent or the value of a close delimiter property.
<when>
The first character of the delimiter was entered with a named
character reference.
```

```
<propdef rcsnm=role string strnorm=rcsgener clause="96001 FIG30">
<desc>
The name of the delimiter role.

<propdef optional rcsnm=origdelm appnm="original delim"
fullnm="original delimiter" string clause="92102 FIG22">
<desc>
The delimiter as originally entered before any upper-case substitution.

<classdef rcsnm=name clause="93001">
<desc>
A name within markup.
<note>
Names in attribute values are represented by nodes of type attvaltk
rather than name.

<propdef rcsnm=origname appnm="original name" string clause="93005">
<desc>
The characters of the name as originally entered before
any upper-case substitution.

<classdef rcsnm=rname appnm="reserved name" clause="d4701">
<desc>
A token in markup that is recognized as a reserved name.

<propdef rcsnm=refname appnm="ref name" fullnm="reference name"
string strnorm=rcsgener clause="d4704">
<desc>
The reference reserved name.

<propdef optional rcsnm=origname appnm="original name" string
clause="93005">
<desc>
The reserved name as originally entered before any upper-case
substitution.

<classdef rcsnm=literal conprop=value clause="a1201 79302 a1701 a1603">
<desc>
A parameter literal, attribute value literal, minimum literal, or
system identifier.

<propdef subnode optional rcsnm=open appnm="open delim"
fullnm="open delimiter" node ac=gendelm clause="96100 FIG30">

<propdef subnode rcsnm=value nodelist
ac="entstart entend datachar sdata intignch"
clause="a1202 91001 a1702 80002">
<desc>
Interpreted value of literal.
<note>
If the literal is an attribute value literal for a tokenized value,
the value of the literal represents the attribute value before
tokenization but after interpretation.

<propdef subnode optional rcsnm=close appnm="close delim"
fullnm="close delimiter" node ac=gendelm clause="96100 FIG30">
```

```
<classdef rcsnm=number clause="93002">
<desc>
A number in markup that is not a character number in
a character reference.
<note>
Numbers in attribute values are represented by nodes of type attvaltk
rather than number.

<propdef rcsnm=digits string  clause="93002">

<classdef rcsnm=crefcnum appnm="char ref char number"
fullnm="character reference character number" clause="95001">
<desc>
A character number occurring in a character reference.
<note>
The numeric value of the number is determined by the char property of
the datachar node.

<propdef optional rcsnm=ndigits appnm="n digits" fullnm="number of digits"
integer clause="95003 93002">
<desc>
The number of digits used to specify the value.

<classdef rcsnm=refendre appnm="ref end re" fullnm="reference end RE"
clause="94502">
<desc>
An RE in markup that is used as a reference end.

<classdef rcsnm=attvalue appnm="attribute value" clause="79400">
<desc>
An attribute value specification that is an attribute value
rather than an attribute value literal.
<note>
Do not confuse this with the attasgn class.

<propdef rcsnm=value string clause="93005">
<desc>
The value before any upper-case substitution.

<classdef rcsnm=nmtoken appnm="name token" clause="93003">
<desc>
A name token in markup.
<note>
This is used only for name tokens in name token groups in
declared values. Name tokens in attribute values are represented by
nodes of type attvaltk rather than nmtoken.

<propdef rcsnm=origname appnm="original name token" string
clause="93005">
<desc>
The characters of the name token as originally entered before
any upper-case substitution.

<classdef rcsnm=msstart appnm="marked section start"
fullnm="marked section declaration start" conprop=markup clause="a4002">
```

```
<desc>
The part of a marked section declaration preceding the marked section.

<propdef subnode optional rcsnm=markup nodelist
ac="gendelm rname ssep entstart entend comment ignmrkup" clause="a4002">
<note>
First child will be gendelm for mdo, last will be gendelm for
dso.

<propdef rcsnm=status enum clause="a4201">
<desc>
Effective status of marked section.

<enumdef rcsnm=ignore>
<enumdef rcsnm=cdata>
<enumdef rcsnm=rcdata>
<enumdef rcsnm=include>
<enumdef rcsnm=temp>

<classdef rcsnm=msend appnm="marked section end" conprop=markup
clause="a4003">

<propdef subnode optional rcsnm=markup nodelist ac=gendelm
clause="FIG3e FIG3h">
<note>
Will be a gendelm for the msc and a gendelm for the mdc.

</psmodule>

<!-- SGML Declaration-related abstract classes and properties -->

<psmodule rcsnm=sdclabs fullnm="sgml declaration abstract" dependon=baseabs>

<propdef rcsnm=sgmlver appnm="sgml version" string
cn=sgmldoc clause="d0002">
<desc>
The minimum literal specified as the first parameter of the SGML
declaration applicable to this document or subdocument.

<propdef subnode rcsnm=docchset appnm="document char set"
fullnm="document character set" node ac=charset cn=sgmldoc
clause="d1001">

<propdef subnode rcsnm=capset appnm="capacity set" node
ac=capset cn=sgmldoc clause="d2001">

<propdef rcsnm=synscope appnm="syntax scope"
fullnm="concrete syntax scope" enum cn=sgmldoc clause="d3002">

<enumdef rcsnm=instance>
<enumdef rcsnm=document>

<propdef subnode rcsnm=dclsyn appnm="decl syntax"
fullnm="declared concrete syntax" node ac=syntax cn=sgmldoc
clause="d4001">
```

```
<propdef subnode rcsnm=refsyn appnm="ref syntax"
fullnm="reference concrete syntax" node ac=syntax cn=sgmldoc
clause="d4002 e0001 FIG70">
<desc>
The reference concrete syntax used for the SGML declaration and,
if the concrete syntax scope is INSTANCE, the prolog.
<note>
Not a property of sgmlcsts because it depends on the document character
set.

<propdef irefnode rcsnm=prosyn appnm="prolog syntax"
fullnm="prolog concrete syntax" node ac=syntax cn=sgmldoc
clause="d4001">
<desc>
The concrete syntax for the prolog.

<propdef subnode rcsnm=features fullnm="feature use" node
ac=features cn=sgmldoc clause="d5001">

<classdef rcsnm=charset appnm="char set" fullnm="character set"
conprop=chdescs clause="d1000">

<propdef subnode rcsnm=chdescs appnm="char descs"
fullnm="character descriptions" nodelist ac=chardesc
clause="d1101">

<classdef rcsnm=chardesc appnm="char desc" fullnm="character description"
clause="d1122">

<propdef rcsnm=descnum appnm="desc set number"
fullnm="described set character number" integer clause="d1123">

<propdef rcsnm=nchars appnm="n chars" fullnm="number of characters"
integer clause="d1125">

<propdef rcsnm=basenum appnm="base set number"
fullnm="base set character number" integer clause="d1124">
<when>
Character description included a base set character number.

<propdef rcsnm=baseset appnm="base char set" fullnm="base character set"
string  clause="d1111">
<desc>
The public identifier of the base character set.
<when>
Character description included a base set character number.

<propdef rcsnm=desclit appnm="desc literal"
fullnm="description literal" string
clause="a1701">
<when>
Character description not entered as base set number.

<classdef rcsnm=syntax fullnm="concrete syntax" clause="d4000">
<note>
This represents a concrete syntax bound to this document's document
```

```
character set. Characters are characters in the document character set
not in the syntax reference character set.

<propdef rcsnm=shunctrl appnm="shunchar controls" boolean
clause="d4204">
<desc>
True if SHUNCHAR included CONTROLS.

<propdef rcsnm=shunchar fullnm="shunned character numbers"
intlist clause="d4201">

<propdef subnode rcsnm=synchset appnm="syntax ref char set"
fullnm="syntax-reference character set" node ac=charset
clause="d4301">

<propdef rcsnm=re fullnm="record end" char clause="d4401">

<propdef rcsnm=rs fullnm="record start" char clause="d4401">

<propdef rcsnm=space char clause="d4401">

<propdef subnode rcsnm=addfuns appnm="added function chars"
fullnm="added function characters" nmndlist ac=addfun
acnmprop=name clause="d4401">

<propdef rcsnm=lcnmstrt string clause="d4503">

<propdef rcsnm=ucnmstrt string clause="d4504">

<propdef rcsnm=lcnmchar string clause="d4505">

<propdef rcsnm=ucnmchar string clause="d4506">

<propdef rcsnm=substgen appnm="subst general names"
fullnm="substitute general names" boolean clause="d4507">
<desc>
True if GENERAL YES is specified in NAMECASE.

<propdef rcsnm=substent appnm="subst entity names"
fullnm="substitute entity names" boolean clause="d4507">
<desc>
True if ENTITY YES is specified in NAMECASE.

<propdef subnode rcsnm=gdasns appnm="general delim assocs"
fullnm="general delimiter role associations"
nmndlist ac=dlmrlas acnmprop=role clause="d4611">
<desc>
There is a term for every general delimiter role whether or not
it is changed from that prescribed by the reference concrete syntax.
The terms occur in alphabetical order of their (abstract-syntax)
role names.

<propdef rcsnm=srdelms appnm="shortref delims"
fullnm="short reference delimiters" strlist clause="d4621">

<propdef subnode rcsnm=slitasns appnm="syntax literal assocs"
```

```
fullnm="syntax literal associations" nmndlist ac=synlitas
acnmprop=resname clause="d4701">
<desc>
The syntax literal/reserved name associations specified by the concrete
syntax. There is a term for every reserved name whether or not
it is changed from that prescribed by the reference concrete syntax.
The terms occur in alphabetical order of the syntactic literals.

<propdef rcsnm=attcnt integer clause="FIG41">
<propdef rcsnm=attsplen integer clause="FIG42">
<propdef rcsnm=bseqlen integer clause="FIG43">
<propdef rcsnm=dtaglen integer clause="FIG44">
<propdef rcsnm=dtemplen integer clause="FIG45">
<propdef rcsnm=entlvl integer clause="FIG46">
<propdef rcsnm=grpcnt integer clause="FIG47">
<propdef rcsnm=grpgtcnt integer clause="FIG48">
<propdef rcsnm=grplvl integer clause="FIG49">
<propdef rcsnm=litlen integer clause="FIG4a">
<propdef rcsnm=namelen integer clause="FIG4b">
<propdef rcsnm=normsep integer clause="FIG4c">
<propdef rcsnm=pilen integer clause="FIG4d">
<propdef rcsnm=taglen integer clause="FIG4e">
<propdef rcsnm=taglvl integer clause="FIG4f">

<classdef rcsnm=addfun appnm="added function char"
fullnm="added function character" clause="d4400">

<propdef rcsnm=name string strnorm=general
clause="d4402">

<propdef rcsnm=class fullnm="function class" enum clause="d4403">
<enumdef rcsnm=funchar>
<enumdef rcsnm=msichar>
<enumdef rcsnm=msochar>
<enumdef rcsnm=msschar>
<enumdef rcsnm=sepchar>

<propdef rcsnm=char fullnm=character char clause="95003">
<desc>
Character assigned to function.

<classdef rcsnm=dlmrlas appnm="delim role assoc"
fullnm="delimiter role association" clause="d4610">
<desc>
The association, made by a concrete syntax, of a character string with
an abstract-syntax delimiter role.

<propdef rcsnm=role string strnorm=rcsgener clause="d4612">
<desc>
The name of the role.

<propdef rcsnm=delm appnm=delim fullnm=delimiter string
strnorm=general clause="d4611">
<desc>
The string to be used in the document.
```

```
<classdef rcsnm=synlitas appnm="syntactic literal assoc"
fullnm="syntactic literal association" clause="d4700">
<desc>
The association, made by a concrete syntax, of a reserved name with
an abstract-syntax syntactic literal.

<propdef rcsnm=synlit appnm="syntactic literal"
string strnorm=rcsgener clause="d4702">
<desc>
The syntactic literal.  (More precisely, the name which when enclosed in
double quotation marks becomes the syntactic literal.)

<propdef rcsnm=resname appnm="reserved name" string
strnorm=general clause="d4702">
<desc>
The reserved name to be used in the document.
<note>
In the reference concrete syntax, the syntactic literal is
identical to the reserved name.

<classdef rcsnm=capset appnm="capacity set" clause="d2000">

<propdef rcsnm=totalcap integer clause="FIG51">
<propdef rcsnm=entcap integer clause="FIG52">
<propdef rcsnm=entchcap integer clause="FIG53">
<propdef rcsnm=elemcap integer clause="FIG54">
<propdef rcsnm=grpcap integer clause="FIG55">
<propdef rcsnm=exgrpcap integer clause="FIG56">
<propdef rcsnm=exnmcap integer clause="FIG57">
<propdef rcsnm=attcap integer clause="FIG58">
<propdef rcsnm=attchcap integer clause="FIG59">
<propdef rcsnm=avgrpcap integer clause="FIG5a">
<propdef rcsnm=notcap integer clause="FIG5b">
<propdef rcsnm=notchcap integer clause="FIG5c">
<propdef rcsnm=idcap integer clause="FIG5d">
<propdef rcsnm=idrefcap integer clause="FIG5e">
<propdef rcsnm=mapcap integer clause="FIG5f">
<propdef rcsnm=lksetcap integer clause="FIG5g">
<propdef rcsnm=lknmcap integer clause="FIG5h">

<classdef rcsnm=features fullnm="feature use" clause="d5000">

<propdef rcsnm=datatag boolean clause="d5101">
<desc>
True if DATATAG is YES.

<propdef rcsnm=omittag boolean clause="d5101">
<desc>
True if OMITTAG is YES.

<propdef rcsnm=rank boolean clause="d5101">
<desc>
True if RANK is YES.

<propdef rcsnm=shorttag boolean clause="d5101">
<desc>
```

```
True if SHORTTAG is YES.

<propdef rcsnm=simple integer clause="d5201">
<desc>
0 if SIMPLE is NO.

<propdef rcsnm=implicit boolean clause="d5201">
<desc>
True if IMPLICIT is YES.

<propdef rcsnm=explicit integer clause="d5201">
<desc>
0 if EXPLICIT is NO.

<propdef rcsnm=concur integer clause="d5301">
<desc>
0 if CONCUR is NO.

<propdef rcsnm=subdoc integer clause="d5301">
<desc>
0 if SUBDOC is NO.

<propdef rcsnm=formal boolean clause="d5301">
<desc>
True if FORMAL is YES.

</psmodule>

<!-- SGML Declaration-related SGML document string classes and properties -->

<psmodule rcsnm=sdclsds fullnm="SGML declaration SGML document string"
dependon=basesds1>

<propdef subnode optional rcsnm=sgmldcl appnm="sgml decl"
fullnm="SGML declaration" node ac=sgmldcl cn=sgmldoc
clause="d0001">
<when>
SGML declaration was explicitly present.

<propdef rcsnm=sdcltype appnm="sgml decl type"
fullnm="SGML declaration type" enum cn=sgmldoc clause="62300">

<enumdef rcsnm=explicit>
<desc>
The SGML declaration was explicitly specified.

<enumdef rcsnm=implied>
<desc>
The SGML declaration was implied.

<enumdef rcsnm=inherit>
<desc>
The SGML declaration comes from the SGML document of which
this is a subdocument.

<classdef rcsnm=sgmldcl appnm="sgml decl" fullnm="SGML declaration"
```

```
conprop=markup clause="d0000">

<propdef subnode rcsnm=markup nodelist
ac="ssep comment name number rname literal gendelm" clause="d0001">
<note>
Also includes any s separators before the SGML declaration;
last child is gendelm for mdc delimiter.

</psmodule>

<!-- Prolog-related abstract classes and properties, level 1 -->

<psmodule rcsnm=prlgabs1 fullnm="prolog abstract level 1"
dependon=prlgabs0>

<propdef subnode rcsnm=attdefs appnm="attribute defs"
fullnm="attribute definitions" nmndlist ac=attdef acnmprop=name
cn=notation clause="b3002">

<propdef irefnode rcsnm=attdef appnm="attribute def"
fullnm="attribute definition" node ac=attdef cn=attasgn
clause="b3003">

<propdef irefnode rcsnm=elemtype appnm="element type" node ac=elemtype
cn=element clause="b2101">

<propdef subnode rcsnm=dfltent appnm="default entity" node ac=dfltent
clause="a5105" cn=doctype>
<when>
The DTD declared a default for undeclared entity names.  (Each such
undeclared name is associated with an entity using this node as
a pattern, but in certain cases, the system may not select the
same entity for each name.)

<propdef subnode rcsnm=elemtps appnm="element types" nmndlist
ac="elemtype rankstem" acnmprop="gi stem" cn=doctype clause="b2101" >
<desc>
Generic identifiers or rank stems used to name elements.

<propdef subnode rcsnm=parments appnm="parameter entities"
nmndlist ac=entity acnmprop=name cn=doctype
clause="b1004" >
<note>
Includes entities not explicitly declared, as discussed above in
the description of this class.

<classdef rcsnm=elemtype appnm="element type"
fullnm="element type definition" clause="b2000">

<propdef rcsnm=gi fullnm="generic identifier" string
 strnorm=general clause="78002">

<propdef rcsnm=omitstrt appnm="omit start tag" boolean
clause="b2202">
<desc>
True if start-tag minimization was "O".
```

```
<when>
Element type declaration specified omitted tag minimization.

<propdef rcsnm=omitend appnm="omit end tag" boolean
clause="b2203">
<desc>
True if end-tag minimization was "O".
<when>
Element type declaration specified omitted tag minimization.

<propdef rcsnm=contype appnm="content type" enum clause="b2300">

<enumdef rcsnm=cdata>
<desc>
Declared content of CDATA.

<enumdef rcsnm=rcdata>
<desc>
Declared content of RCDATA.

<enumdef rcsnm=empty>
<desc>
Declared content of EMPTY.

<enumdef rcsnm=any>
<desc>
Content model of ANY.

<enumdef rcsnm=modelgrp appnm="model group">
<desc>
Content model that is a model group.

<propdef subnode rcsnm=modelgrp appnm="model group" node
ac=modelgrp clause="b2402">
<when>
Element type declaration includes content model that has a model group.

<propdef rcsnm=excls appnm=exclusions strlist clause="b2521">
<when>
Contype is any or modelgrp.

<propdef rcsnm=incls appnm=inclusions strlist clause="b2511">
<when>
Contype is any or modelgrp.

<propdef subnode rcsnm=attdefs appnm="attribute defs"
fullnm="attribute definitions" nmndlist ac=attdef acnmprop=name
clause="b3003">

<classdef rcsnm=modelgrp appnm="model group" conprop=tokens
clause="b2402">
<desc>
A model group or a data tag group.
<note>
A data tag group is represented by a model group node with connector
equal to seq whose first token is an elemtk and whose second token
```

```
is a pcdatatk.

<propdef rcsnm=connect appnm=connector enum clause="b2410">
<desc>
Connector used within model group.

<enumdef rcsnm=and>
<enumdef rcsnm=or>
<enumdef rcsnm=seq>

<propdef rcsnm=occur appnm="occur indicator" fullnm="occurrence indicator"
enum clause="b2420">
<when>
Model group has an occurrence indicator.

<enumdef rcsnm=opt>
<enumdef rcsnm=plus>
<enumdef rcsnm=rep>

<propdef subnode rcsnm=tokens appnm="content tokens" nodelist
ac="modelgrp pcdatatk elemtk" clause="b2403">

<classdef rcsnm=pcdatatk appnm="pcdata token" clause="b2404">

<classdef rcsnm=elemtk appnm="element token" clause="b2405">

<propdef rcsnm=gi fullnm="generic identifier" string
 strnorm=general clause="b2405">

<propdef rcsnm=occur appnm="occur indicator" fullnm="occurrence indicator"
enum clause="b2405">
<when>
Element token has an occurrence indicator.

<enumdef rcsnm=opt>
<enumdef rcsnm=plus>
<enumdef rcsnm=rep>

<classdef rcsnm=attdef appnm="attribute def" fullnm="attribute definition"
conprop=dfltval clause="b3003">

<propdef rcsnm=name string strnorm=general
clause="b3201">

<propdef rcsnm=dcltype appnm="decl value type"
fullnm="declared value prescription type" enum clause="b3301">

<enumdef rcsnm=cdata>
<enumdef rcsnm=entity>
<enumdef rcsnm=entities>
<enumdef rcsnm=id>
<enumdef rcsnm=idref>
<enumdef rcsnm=idrefs>
<enumdef rcsnm=name>
<enumdef rcsnm=names>
<enumdef rcsnm=nmtoken>
```

```
<enumdef rcsnm=nmtokens>
<enumdef rcsnm=number>
<enumdef rcsnm=numbers>
<enumdef rcsnm=nutoken>
<enumdef rcsnm=nutokens>
<enumdef rcsnm=notation>
<enumdef rcsnm=nmtkgrp appnm="name token group">
<desc>
The declared value was a name token group.

<propdef rcsnm=tokens strlist clause="b3301">
<desc>
A list of strings specifying the allowed tokens.
<when>
Declared value is a name token group or a notation.

<propdef rcsnm=dflttype appnm="default value type" enum
clause="b3401">

<enumdef rcsnm=value>
<desc>
The default value was an attribute value specification without #FIXED.

<enumdef rcsnm=fixed>
<enumdef rcsnm=required>
<enumdef rcsnm=current>
<enumdef rcsnm=conref>
<enumdef rcsnm=implied>

<propdef subnode rcsnm=dfltval appnm="default value" nodelist
ac="attvaltk datachar sdata intignch entstart entend" clause="b3409">
<when>
The default value includes an attribute value specification.

<propdef irefnode rcsnm=curgrp appnm="current group" nodelist
ac=attdef clause="b3001">
<desc>
All the attdef nodes that represent the same attribute definition
and which will therefore share the same current value.
<note>
There will be as many members as there were associated element types
in the attribute definition list declaration
that declared this attribute definition.
<when>
The default value type is CURRENT.

<propdef rcsnm=curattix appnm="current attribute index" integer
clause="b3001">
<desc>
The number of preceding attribute definitions in the document type
declaration with a default value type of CURRENT.
<note>
All the attdef nodes in the value of the curgrp property of an attdef
node will exhibit the same value for the curattix property.
Two attdef nodes will share the same current value just in case they
exhibit the same value for the curattix property.
```

```
<when>
The default value type is CURRENT.

<classdef rcsnm=dfltent appnm="default entity">

<propdef rcsnm=enttype appnm="entity type" enum clause="a5502">

<enumdef rcsnm=text fullnm="SGML text">
<enumdef rcsnm=cdata>
<enumdef rcsnm=sdata>
<enumdef rcsnm=ndata>
<enumdef rcsnm=subdoc appnm=subdocument>
<enumdef rcsnm=pi>

<propdef rcsnm=text string fullnm="replacement text"
clause="92101">
<when>
The default entity declaration declares an internal entity.

<propdef subnode rcsnm=extid appnm="external id"
fullnm="external identifier" node ac=extid clause="a1601">
<when>
The default entity declaration declares an external entity.

<propdef subnode rcsnm=atts appnm=attributes
nmndlist ac=attasgn acnmprop=name clause="b4120">
<desc>
A list of data attribute assignments, one for each declared attribute of the
entity in the order in which they were declared in the attribute
definition list declaration.
<when>
The default entity declaration declares an external entity.

<propdef rcsnm=notname appnm="notation name" string
strnorm=general clause="79408">
<when>
The default entity declaration declares an external entity.

<propdef irefnode rcsnm=notation node ac=notation clause="b4001">
<when>
The default entity declaration declares an external entity.

</psmodule>

<!-- Prolog-related SDS classes and properties -->

<psmodule rcsnm=prlgsds fullnm="prolog SGML document string"
dependon=basesds1>

<propdef irefnode rcsnm=entdcl appnm="entity decl"
fullnm="entity declaration" node ac=entdcl cn=entity
clause="a5001">

<propdef irefnode rcsnm=entdcl appnm="entity decl"
fullnm="entity declaration" node ac=entdcl cn=dfltent
clause="a5001">
```

```
<propdef irefnode rcsnm=notdcl appnm="notation decl"
fullnm="notation declaration" node ac=notdcl cn=notation
clause="b4001">

<propdef irefnode rcsnm=attdldcl appnm="attribute def list decl"
fullnm="attribute definition list declaration" nodelist ac=attdldcl
cn=notation clause="b4111">
<when>
The notation has at least one associated ATTLIST.

<propdef irefnode rcsnm=eltpdcl appnm="element type decl"
fullnm="element type declaration" node ac=eltpdcl cn=elemtype
clause="b2001">

<propdef irefnode rcsnm=attdldcl appnm="attribute def list decl"
fullnm="attribute definition list declaration"
nodelist ac=attdldcl cn=elemtype clause="b3001">
<when>
The element type has at least one associated ATTLIST declaration.

<propdef irefnode rcsnm=doctpdcl appnm="document type decl"
fullnm="document type declaration" node ac=doctpdcl
cn=doctype clause="b1001">

<propdef irefnode rcsnm=attvalsp appnm="attribute value spec"
fullnm="attribute value specification"
node ac="attvalue literal" cn=attdef clause="79002">
<when>
Default value includes attribute value specification.

<classdef rcsnm=doctpdcl appnm="document type decl"
fullnm="document type declaration" mayadd clause="b1000">

<propdef subnode rcsnm=markup nodelist
ac="ssep comment name rname literal msstart msend msignch entstart
    entend comdcl pi eltpdcl entdcl notdcl attdldcl usemap srmapdcl"
    clause="b1001">
<note>
First child is gendelm for mdo delimiter; last is gendelm
for mdc delimiter. If there is an external entity, its entend node
will appear immediately before the gendelm for the dsc delimiter,
if there is one, and otherwise immediately before the gendelm node
for the mdc delimiter.

<propdef irefnode rcsnm=doctype appnm="document type" node
ac=doctype clause="b1008">

<propdef subnode rcsnm=entity node ac=entity clause="b1008">
<when>
Document type declaration includes external identifier.

<classdef rcsnm=attdldcl appnm="attribute def list decl"
fullnm="attribute definition list declaration" mayadd clause="b3000">

<propdef subnode rcsnm=markup nodelist
```

```
ac="ssep comment entstart entend gendelm name nmtoken attvalue literal"
clause="b3001">

<propdef irefnode rcsnm=asseltps appnm="assoc element types"
fullnm="associated element types" nodelist ac=elemtype
clause="b3001">
<desc>
The element types to which the attribute definition list is applicable,
ordered as their names occur in the attribute definition
list declaration. This does not include undefined element types.

<propdef irefnode rcsnm=assnots appnm="assoc notations"
fullnm="associated notations" nodelist ac=notation clause="b3001">

<classdef rcsnm=eltpdcl appnm="element type decl"
fullnm="element type declaration" mayadd clause="b2000">

<propdef subnode rcsnm=markup nodelist
ac="ssep comment entstart entend gendelm name number" clause="b2001">

<propdef irefnode rcsnm=elemtype appnm="element type"
fullnm="element type" node ac=elemtype clause="b2101">

<classdef rcsnm=entdcl appnm="entity decl" fullnm="entity declaration"
mayadd clause="a5000">
<desc>
An entity declaration that is not ignored.

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment gendelm name rname literal attvalue"
clause="a5001">

<propdef subnode rcsnm=entity node ac=entity clause="a5201">
<desc>
The entity declared by the entity declaration.
<when>
The entity declaration does not declare the default entity.

<classdef rcsnm=notdcl appnm="notation decl"
fullnm="notation declaration" mayadd clause="b4000">

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment literal name rname" clause="b4001">

<propdef irefnode rcsnm=notation node ac=notation clause="b4001">
<desc>
The declared notation.

</psmodule>

<!-- Document instance-related SDS classes and properties -->

<psmodule rcsnm=instsds0 fullnm="instance SGML document string level 0">

<propdef derived rcsnm=included boolean cn=element>
<desc>
```

```
True if and only if the element was an included subelement.

<propdef derived rcsnm=momitend appnm="must omit end tag" boolean
cn=element clause="b2209">
<desc>
True if and only if the end tag for the element had to be omitted
because the element had a declared content of empty or
an explicit content reference.

</psmodule>

<psmodule rcsnm=instsds1 fullnm="instance SGML document string level 1"
dependon="instsds0 basesds1">

<!-- Element -->

<propdef subnode optional rcsnm=starttag appnm="start-tag" nodelist
ac="gendelm name ssep entstart entend literal attvalue" cn=element
clause="74001">
<note>
First child is gendelm for stago.
Nodes of type entstart and entend can occur only
in the document type specification.
<when>
A start-tag was specified for the element.

<propdef subnode optional rcsnm=endtag appnm="end-tag" nodelist
ac="gendelm name ssep entstart entend ignmrkup" cn=element clause="75001">
<note>
First child is gendelm for etago or net. Nodes of type entstart,
entend, and ignmrkup can occur only in the document type specification.
<when>
An end-tag (not a data tag) was specified for the element.

<!-- Data character -->

<propdef rcsnm=movedre appnm="moved re" boolean cn=datachar
clause="7610a">
<desc>
True if and only if this character is an RE that was deemed to occur
at a point other than that at which it in fact occurred.
<note>
A node of type repos will indicate the position at which
it in fact occurred.

<propdef irefnode rcsnm=repos appnm="re position" node cn=datachar
ac=repos clause="7610a">
<desc>
The position at which this RE character in fact occurred.
<when>
This character is an RE that was deemed to occur at a point other
than that at which it in fact occurred.

<propdef subnode optional rcsnm=markup nodelist
ac="gendelm name ssep entstart entend refendre shortref" cn=extdata
clause="94401 94402">
```

```
<desc>
The markup of the entity reference.
<note>
ssep, entstart, and entend can occur only in a name group in a named
entity reference.

<classdef rcsnm=ignrs appnm="ignored rs" clause="76101">
<desc>
An RS that was ignored because of the rules in 7.6.1 of ISO 8879.

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" clause="95001">
<when>
The character was the replacement of a named character reference.

<classdef rcsnm=ignre appnm="ignored re" clause="76100">
<desc>
An RE in content that was ignored because of the rules in 7.6.1 of ISO
8879.
<note>
This occurs at the point where the RE originally occurred rather
than at the point it was determined that the RE should be ignored.

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" clause="95001">
<when>
The character was the replacement of a named character reference.

<classdef rcsnm=repos appnm="re position" clause="7610a">
<desc>
The original position of an RE that was deemed by the rules of clause
7.6.1 of ISO 8879 to occur at some point other than that at which it
in fact occurred.
<note>
For each node of type repos, there will be a node of type datachar
with a property movedre that is true.

<propdef irefnode rcsnm=re appnm="record end" node ac=datachar
clause="7610a">
<desc>
The character for which this is the repos.

</psmodule>

<!-- Datatag-related abstract classes and properties -->

<psmodule rcsnm=dtgabs fullnm="datatag abstract" dependon=baseabs>

<propdef derived rcsnm=datatag boolean cn=element clause="73201">
<desc>
True if and only if a data tag served as the end-tag of the element.
<note>
The data characters comprising the data tag will follow the element in
the content of the containing element.
```

```
<propdef rcsnm=dtgtemps appnm="data tag templates" strlist
cn=elemtype clause="b2444">
<when>
The model group was a data tag group.

<propdef rcsnm=dtgptemp appnm="data tag padding template" string
cn=elemtype clause="b2445">
<when>
The model group was a data tag group whose data tag pattern included a
data tag padding template.

</psmodule>

<!-- Rank-related abstract classes and properties -->

<psmodule rcsnm=rankabs fullnm="rank abstract" dependon=prlgabs1>

<propdef derived rcsnm=ranksuff appnm="rank suffix" string
cn=elemtype clause="b2114">
<when>
The element type in the element type declaration included a rank suffix.

<propdef rcsnm=rankstem appnm="rank stem" string cn=elemtype
clause="b2113">
<when>
The element type in the element type declaration used a ranked element
or ranked group.

<propdef rcsnm=rankgrp appnm="rank group" strlist cn=elemtype
clause="b2112">
<desc>
The rank stems in the ranked group.
<when>
The element type declaration included a ranked group.

<classdef rcsnm=rankstem appnm="rank stem" clause="b2113">

<propdef rcsnm=stem string strnorm=general
clause="b2113">
<desc>
Name of rank stem.

<propdef irefnode rcsnm=elemtps appnm="element types"
nodelist ac=elemtype clause="b2112">
<desc>
The element types for which this is a rank stem.

</psmodule>

<!-- Shortref-related abstract classes and properties -->

<psmodule rcsnm=srabs fullnm="shortref abstract" dependon=prlgabs0>

<propdef subnode rcsnm=emptymap appnm="empty short ref map"
fullnm="empty short reference map" node ac=srmap cn=sgmlcsts
```

```
clause="b6004">
<desc>
The empty short reference map.

<propdef subnode rcsnm=srmaps appnm="short ref maps"
fullnm="short reference maps" nmndlist ac=srmap acnmprop=name
cn=doctype clause="b1006">
<note>
Does not include #EMPTY map.

<propdef rcsnm=srmapnm appnm="short ref map name"
fullnm="short reference map name" string
strnorm=general cn=elemtype clause="b6004">
<when>
The element type has an associated short reference map.

<propdef irefnode rcsnm=srmap appnm="short ref map"
fullnm="short reference map" node ac=srmap cn=elemtype
clause="b6101">
<when>
The element type has an associated short reference map.

<classdef rcsnm=srmap appnm="short ref map" fullnm="short reference map"
clause="b5000">

<propdef rcsnm=name string strnorm=general clause="b5002">
<when>
Map is not the implicitly declared #EMPTY map.

<propdef subnode rcsnm=map nmndlist ac=srassoc acnmprop=shortref
clause="b5004">

<classdef rcsnm=srassoc appnm="short ref assoc"
fullnm="short reference association" clause="b5004">

<propdef rcsnm=shortref appnm="short ref"
fullnm="short reference delimiter" string strnorm=general
clause="b5004">

<propdef rcsnm=entname appnm="entity name" string
strnorm=entity clause="b5004">

<propdef irefnode rcsnm=entity node ac=entity clause="b5001">

</psmodule>

<!-- Shortref-related SDS classes and properties -->

<psmodule rcsnm=srsds fullnm="shortref SGML document string"
dependon=basesds1>

<classdef rcsnm=usemap appnm="short ref use decl"
fullnm="short reference use declaration" conprop=markup clause="b6000">

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment gendelm name rname ignmrkup"
```

```
clause="b6001">
<note>
First child is gendelm for mdo delimiter; last is gendelm for mdc
delimiter.

<propdef irefnode rcsnm=asseltps appnm="assoc element types"
fullnm="associated element types" nodelist ac=elemtype
clause="a1501">
<note>
SGML specifies that this does not include element types which had
already been associated with a map.
<when>
The short reference use declaration includes an associated element
type.

<propdef irefnode rcsnm=srmap node ac=srmap clause="b6002">

<classdef rcsnm=shortref appnm="short ref"
fullnm="short reference delimiter" clause="e4620">

<propdef rcsnm=origdelm appnm="original delim"
fullnm="original delimiter" string clause="96601">
<desc>
The short reference delimiter as originally entered.

<propdef subnode optional rcsnm=namecref appnm="named char ref"
fullnm="named character reference" nodelist
ac="gendelm name refendre" clause="95001">
<when>
The first character of the delimiter was entered with a named
character reference.

<classdef rcsnm=srmapdcl appnm="short ref map decl"
fullnm="short reference mapping declaration" mayadd clause="b5000">

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment gendelm name rname literal"
clause="b5001">
<note>
First child is gendelm for mdo delimiter; last is gendelm for mdc
delimiter.

<propdef irefnode rcsnm=map node ac=srmap clause="b5001">

</psmodule>

<!-- Link-related abstract classes and properties -->

<psmodule rcsnm=linkabs fullnm="link abstract" dependon=prlgabs0>

<propdef subnode rcsnm=emptylks appnm="empty link set" node ac=linkset
cn=sgmlcsts clause="c3004">
<desc>
Empty link set used to disable current link set.

<propdef subnode optional rcsnm=simplelk appnm="simple link info"
```

```
fullnm="simple link information" nmndlist ac=simplelk
acnmprop=linktype cn=element clause="c1431">
<when>
Element is the document element and there are active simple link
processes.

<propdef irefnode rcsnm=linkatts appnm="link attributes"
nmndlist ac=attasgn acnmprop=name cn=element clause="c1402">
<desc>
A list of attribute assignments, one for each declared link attribute
of the element.
<note>
The origin of the link attributes will be the link rule.

<propdef derived rcsnm=rsltgi appnm="result gi"
fullnm="result element generic identifier" string
strnorm=general cn=element clause="c2202">
<when>
There is an applicable link rule which is an explicit link rule whose
result element is not implied.

<propdef irefnode rcsnm=rsltelem appnm="result element type"
node ac=elemtype cn=element clause="c2202">
<when>
There is an applicable link rule which is an explicit link rule whose
result element is not implied.

<propdef irefnode rcsnm=rsltatts appnm="result attributes"
nmndlist ac=attasgn acnmprop=name cn=element clause="c2203">
<note>
The origin of the attributes will be the link rule.
<when>
There is an applicable link rule which is an explicit link rule whose
result element is not implied.

<propdef irefnode rcsnm=lksetinf appnm="link set info"
fullnm="link set information" nodelist ac=linkrule cn=element
clause="c2205">
<desc>
Link rules in the current link set whose source element type is implied.
<when>
There is an active explicit link process.

<propdef irefnode rcsnm=lksetinf appnm="link set info"
fullnm="link set information" nodelist ac=linkrule cn=datachar>
<desc>
Link rules in the current link set whose source element type is implied.
<when>
There is an active explicit link process and the character occurs
in content.

<classdef rcsnm=simplelk appnm="simple link info"
fullnm="simple link information" clause="c1430">

<propdef rcsnm=linktype appnm="link type" string
strnorm=general clause="c1001">
```

```
<desc>
The link type name of the simple link process.

<propdef subnode rcsnm=atts appnm=attributes
nmndlist ac=attasgn acnmprop=name clause="c1402">

<classdef rcsnm=linktype appnm="link type">

<propdef rcsnm=name string strnorm=general
clause="c1002">

<propdef rcsnm=active boolean>
<desc>
True if and only if link type is active.

<propdef rcsnm=ltkind appnm="link type kind"
fullnm="kind of link type" enum clause="c1001">
<enumdef rcsnm=simple>
<enumdef rcsnm=implicit>
<enumdef rcsnm=explicit>

<propdef rcsnm=srcname appnm="source document type name" string
 strnorm=general clause="c1302">

<propdef irefnode rcsnm=source appnm="source document type" node
ac=doctype clause="c1305 c1306">
<note>
For a simple link type, this will always be the base document type.

<propdef rcsnm=rsltname appnm="result document type name" string
 strnorm=general clause="c1303">

<propdef irefnode rcsnm=result appnm="result document type" node
ac=doctype clause="c1306">
<when>
The link type is an explicit link type.

<propdef subnode rcsnm=inilkset appnm="initial link set" node
ac=linkset clause="c2004">
<when>
The link type is not simple.

<propdef subnode rcsnm=idlkset appnm="id link set" node ac=linkset
clause="c2300">
<when>
The link type declaration subset includes an ID link set declaration.

<propdef subnode rcsnm=linksets appnm="link sets" nmndlist
ac=linkset acnmprop=name clause="c1401">
<note>
Does not include #INITIAL or #EMPTY or ID link set.

<classdef rcsnm=linkset appnm="link set" conprop=lkrules clause="c2000">

<propdef rcsnm=name string strnorm=general
clause="c2003">
```

```
<when>
Link set is not #INITIAL nor #EMPTY nor the ID link set.

<propdef subnode rcsnm=lkrules appnm="link rules" nodelist
ac=linkrule clause="c2002">

<classdef rcsnm=linkrule appnm="link rule" clause="c2002">

<propdef rcsnm=assgis appnm="assoc gis"
fullnm="associated generic identifiers" strlist
clause="c2101">
<desc>
The names of the associated element types.
<when>
The link rule is not an explicit link rule whose source element type
is implied.

<propdef irefnode rcsnm=asseltps appnm="assoc element types"
fullnm="associated element types" nodelist ac=elemtype
clause="c2101">
<when>
The link rule is not an explicit link rule whose source element type
is implied.

<propdef rcsnm=id fullnm="unique identifier" string
strnorm=general clause="c2301">
<when>
Link rule occurs in ID link set declaration.

<propdef irefnode rcsnm=uselink node ac=linkset clause="c2104">
<when>
The link rule includes a USELINK parameter.

<propdef rcsnm=uselknm appnm="uselink name" string
strnorm=general clause="c2104">
<desc>
The link set named by the USELINK parameter.
<when>
The link rule includes a USELINK parameter.

<propdef derived rcsnm=postlkrs appnm="postlink restore" boolean
clause="c2101">
<desc>
True if the link rule includes a POSTLINK parameter of #RESTORE.

<propdef irefnode rcsnm=postlkst appnm="postlink set" node
ac=linkset clause="c2101">
<when>
The link set specification did not specify #RESTORE.

<propdef rcsnm=postlknm string strnorm=general
clause="c2101">
<desc>
The token specified for the link set specification following POSTLINK.
<when>
The link rule includes a POSTLINK parameter.
```

```
<propdef subnode rcsnm=linkatts appnm="link attributes"
nmndlist ac=attasgn acnmprop=name clause="c2102">
<when>
The link rule is not an explicit link rule whose source element type
is implied.

<propdef rcsnm=rsltgi appnm="result gi"
fullnm="result element generic identifier" string
strnorm=general clause="c2202">
<when>
The link rule is an explicit link rule whose result element type is
not implied.

<propdef irefnode rcsnm=rsltelem appnm="result element type" node
ac=elemtype clause="c2202">
<when>
The link rule is an explicit link rule whose result element type is
not implied.

<propdef subnode rcsnm=rsltatts appnm="result attributes"
nmndlist ac=attasgn acnmprop=name clause="c2203">
<when>
The link rule is an explicit link rule whose result element type is
not implied.

</psmodule>

<!-- Link-related SDS classes and properties -->

<psmodule rcsnm=linksds fullnm="link SGML document string"
dependon=basesds1>

<propdef irefnode rcsnm=lksetdcl appnm="link set decl"
fullnm="link set declaration" node ac="lksetdcl idlkdcl"
cn=linkset clause="c2001">
<when>
Link set is not #EMPTY.

<propdef irefnode rcsnm=lktpdcl appnm="link type decl"
fullnm="link type declaration" node ac=lktpdcl cn=linktype
clause="c1001">

<classdef rcsnm=lktpdcl appnm="link type decl" fullnm="link type declaration"
mayadd clause="c1000">

<propdef subnode rcsnm=markup nodelist
ac="ssep comment name rname literal msstart msignch msend
    entstart entend pi comdcl entdcl attdldcl lksetdcl idlkdcl"
    clause="c1001">

<propdef irefnode rcsnm=linktype appnm="link type" node
ac=linktype>

<propdef subnode rcsnm=entity node ac=entity clause="c1004">
<when>
```

```
Link type definition includes external identifier.

<classdef rcsnm=lksetdcl appnm="link set decl" fullnm="link set declaration"
mayadd clause="c2000">

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment gendelm name rname literal attvalue"
clause="c2001">

<propdef irefnode rcsnm=linkset appnm="link set" node
ac=linkset clause="c2001">

<classdef rcsnm=idlkdcl appnm="id link set decl"
fullnm="ID link set declaration" mayadd clause="c2300">

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment gendelm name rname literal attvalue"
clause="c2301">

<propdef irefnode rcsnm=linkset appnm="link set" node ac=linkset
clause="c2301">

<classdef rcsnm=uselink appnm="link set use decl"
fullnm="link set use declaration" conprop=markup clause="c3000">
<desc>
A link set use declaration that is not ignored.

<propdef subnode rcsnm=markup nodelist
ac="entstart entend ssep comment gendelm name rname ignmrkup"
clause="c3001">
<note>
First child is gendelm for mdo delimiter; last is gendelm
for mdc delimiter.

<propdef derived rcsnm=restore boolean clause="c3002">
<desc>
True if the link set specification specified #RESTORE.

<propdef irefnode rcsnm=linkset node ac=linkset clause="c3002">
<when>
The link set specification did not specify #RESTORE.

<propdef rcsnm=lksetnm string strnorm=general
clause="c3002">
<desc>
The token specified for the link set specification.

<propdef rcsnm=linktpnm appnm="link type name" string
 strnorm=general clause="c3001">

<propdef irefnode rcsnm=linktype appnm="link type" node
ac=linktype clause="c3001">

</psmodule>

<!-- Subdoc-related abstract classes and properties -->
```

```
<psmodule rcsnm=subdcabs fullnm="subdoc abstract" dependon=baseabs>

<classdef rcsnm=subdoc appnm=subdocument fullnm="reference to subdocument">
<desc>
The result of referencing a subdocument entity.

<propdef rcsnm=entname appnm="entity name" string
strnorm=entity clause="a5101">

<propdef irefnode rcsnm=entity node ac=entity clause="c5501">

</psmodule>

<!-- Subdoc-related SDS classes and properties -->

<psmodule rcsnm=subdcsds fullnm="subdoc SGML document string"
dependon="basesds1 subdcabs">

<propdef subnode optional rcsnm=markup nodelist
ac="gendelm name ssep entstart entend refendre shortref" cn=subdoc
clause="94401">
<desc>
The markup of the entity reference.
<note>
ssep, entstart, and entend can occur only in a name group in a named
entity reference.

</psmodule>

<!-- Formal public identifier-related abstract classes and properties -->

<psmodule rcsnm=fpiabs fullnm="formal public identifier abstract"
dependon=baseabs>

<propdef subnode optional rcsnm=fpi appnm="formal public id"
fullnm="formal public identifier" node ac=fpi cn=extid
clause="a2001">
<when>
FORMAL YES was specified in the SGML declaration.

<classdef rcsnm=fpi appnm="formal public id" fullnm="formal public identifier"
clause="a2000">
<note>
The string which is the value of each of the string-valued properties
provided by this class is the minimum data specified as such in the
governing productions, without any accompanying "//", "-//", "+//"
or s characters.

<propdef rcsnm=ownertp appnm="owner type" enum clause="a2100">
<desc>
Type of owner identifier.

<enumdef rcsnm=iso>
<enumdef rcsnm=regist appnm=registered>
<enumdef rcsnm=unregist appnm=unregistered>
```

```
<propdef rcsnm=ownerid appnm="owner id" fullnm="owner identifier"
string  clause="a2100">

<propdef rcsnm=textclas appnm="text class" fullnm="public text class"
enum clause="a2210">
<enumdef rcsnm=capacity>
<enumdef rcsnm=charset>
<enumdef rcsnm=document>
<enumdef rcsnm=dtd>
<enumdef rcsnm=elements>
<enumdef rcsnm=entities>
<enumdef rcsnm=lpd>
<enumdef rcsnm=nonsgml>
<enumdef rcsnm=notation>
<enumdef rcsnm=shortref>
<enumdef rcsnm=subdoc>
<enumdef rcsnm=syntax>
<enumdef rcsnm=text>

<propdef rcsnm=unavail appnm=unavailable boolean clause="a2202">
<desc>
True if and only if unavailable text indicator was specified.

<propdef rcsnm=textdesc appnm="text description"
fullnm="public text description" string  clause="a2221">

<propdef rcsnm=textlang appnm="text language"
fullnm="public text language" string clause="a2231">
<when>
The text identifier included a public text language.

<propdef rcsnm=textdseq appnm="text designating sequence"
fullnm="public text designating sequence" string clause="a2241">
<when>
The text identifier included a public text designating sequence.

<propdef rcsnm=textdver appnm="text display version"
fullnm="public text display version" string clause="a2251">
<when>
The text identifier included a public text display version
(that is, there was a // following the public text language
or public text designating sequence).

</psmodule>

<!-- SGML Extended Facilities modules -->

<!-- SGML Architecture abstract classes and properties -->

<psmodule rcsnm=arcabs fullnm="SGML architecture abstract"
dependon="instabs">
<desc>
These properties relate the groves created from architectural
documents to the client documents from which they were derived.
```

```
<!-- Architectural documents -->

<propdef cn=sgmldoc rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this document is an
architectural document of its client document.
<when>
The document is an architectural document.

<propdef cn=sgmldoc rcsnm=arcdocs appnm="arc docs"
fullnm="architectural documents" nmndlist urefnode ac="sgmldoc"
acnmprop="arcname" sd=HyTime clause="A31200">
<desc>
The architectural documents derivable from this document.

<propdef cn=sgmldoc rcsnm=cltdoc appnm="client doc"
fullnm="client document" node urefnode ac="sgmldoc" sd=HyTime
clause="A31000">
<desc>
The document of which this document is an architectural document.
<when>
The document is an architectural document.

<!-- Architectural elements -->

<propdef cn=element rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this element is an
architectural element of its client element.
<when>
The document is an architectural document.

<propdef cn=element rcsnm=arcelems appnm="arc elements"
fullnm="architectural elements" nmndlist urefnode ac="element"
acnmprop="arcname" sd=HyTime clause="A31200">
<desc>
The architectural elements derivable from this element.

<propdef cn=element rcsnm=cltelem appnm="client element"
node urefnode ac="element" sd=HyTime clause="A31000">
<desc>
The element of which this element is an architectural element.
<when>
The element is an architectural element.

<!-- Architectural external data entities -->

<propdef cn=entity rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this entity is an
```

```
architectural entity of its client element.
<when>
The document is an architectural document and the entity is derived
from an external data entity defined in the client document.

<propdef cn=entity rcsnm=arcents appnm="arc entities"
fullnm="architectural entities" nmndlist urefnode ac="entity"
acnmprop="arcname" sd=HyTime clause="A31200 A36200 A36400">
<desc>
The architectural entities derivable from this entity.
<when>
The entity is an external data entity.

<propdef cn=entity rcsnm=cltent appnm="client entity"
node urefnode ac="entity" sd=HyTime clause="A31000">
<desc>
The entity of which this entity is an architectural entity.
<when>
The entity is an architectural external data entity.

<!-- Architectural data characters -->

<propdef cn=datachar rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this data character is an
architectural data character of its client data character.
<when>
The document is an architectural document.

<propdef cn=datachar rcsnm=arcdata appnm="arc data chars"
fullnm="architectural data characters" nmndlist urefnode ac="datachar"
acnmprop="arcname" sd=HyTime clause="A31200">
<desc>
The architectural data characters derivable from this data character.

<propdef cn=datachar rcsnm=cltdata appnm="client data char"
node urefnode ac="datachar" sd=HyTime clause="A31000">
<desc>
The data character of which this data character is an architectural
data character.
<when>
The data character is an architectural data character.

<!-- Architectural system data -->

<propdef cn=sdata rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this system data is
architectural system data of its client system data.
<when>
The document is an architectural document.
```

```
<propdef cn=sdata rcsnm=arcsdata appnm="arc sdata"
fullnm="architectural internal specific character data entity reference result"
nmndlist urefnode ac="sdata" acnmprop="arcname" sd=HyTime clause="A31200">
<desc>
The architectural system data derivable from this system data.

<propdef cn=sdata rcsnm=cltsdata appnm="client sdata"
fullnm="client internal specific character data entity reference result"
node urefnode ac="sdata" sd=HyTime clause="A31000">
<desc>
The system data of which this system data is architectural
system data.
<when>
The system data is architectural system data.

<!-- Architectural external data -->

<propdef cn=extdata rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this external data is
architectural external data of its client external data.
<when>
The document is an architectural document.

<propdef cn=extdata rcsnm=arcexdat appnm="arc external data"
fullnm="architectural external data" nmndlist urefnode ac="extdata"
acnmprop="arcname" sd=HyTime clause="A31200">
<desc>
The architectural external data derivable from this external data.

<propdef cn=extdata rcsnm=cltexdat appnm="client external data"
node urefnode ac="extdata" sd=HyTime clause="A31000">
<desc>
The external data of which this external data is architectural
external data.
<when>
The external data is architectural external data.

<!-- Architectural subdocuments -->

<propdef cn=subdoc rcsnm=arcname appnm="arc name"
fullnm="architecture name" string strnorm=general sd=HyTime
clause="A34100">
<desc>
The name of the architecture for which this subdocument reference is
architectural subdocument reference of its client subdocument reference.
<when>
The document is an architectural document.

<propdef cn=subdoc rcsnm=arcsubds appnm="arc subdocuments"
fullnm="architectural references to subdocuments" nmndlist urefnode
ac="subdoc" acnmprop="arcname" sd=HyTime clause="A31200">
<desc>
The architectural subdocument reference derivable from this
```

```
subdocument reference.

<propdef cn=subdoc rcsnm=cltsubdc appnm="client subdocument"
fullnm="client reference to subdocument" node urefnode ac="subdoc"
sd=HyTime clause="A31000">
<desc>
The subdocument reference of which this subdocument reference is
an architectural subdocument reference.
<when>
The subdocument reference is an architectural subdocument reference.

</psmodule>

<!-- Formal system identifier-related abstract classes and properties -->

<psmodule rcsnm=fsiabs fullnm="formal system identifier abstract"
dependon=baseabs>

<propdef cn=extid rcsnm=fsi appnm="formal system id"
fullnm="formal system identifier" node subnode ac="fsi" optional
sd=HyTime clause="A61000 A64300">
<when>
The system identifier portion of the external identifier is recognized
as a formal system identifier.

<classdef rcsnm=fsi appnm="formal system id"
fullnm="formal system identifier" conprop=sosseqs sd=HyTime
clause="A61000 A64300">

<propdef rcsnm=inclents appnm="included entities" nodelist irefnode
ac="entity" sd=HyTime clause="A66000 A57103">
<desc>
Entities included from notation data, specified using the "included"
FSI entity usage attribute.

<propdef rcsnm=modlevel appnm="mod level" fullnm="modification level"
integer sd=HyTime clause="A66002">
<desc>
The (possibly invalid) modification level of the entity.
<when>
The entity usage tag of the FSI includes a modification level
specification.

<propdef rcsnm=validmod appnm="valid mod level"
fullnm="valid modification level" integer sd=HyTime clause="A66002">
<desc>
The valid modification level of the entity.

<propdef rcsnm=bitcsize appnm="bit combination size" integer sd=HyTime
clause="A66000">

<propdef rcsnm=wordsize appnm="word size"
fullnm="word size and bit-combination ordering" intlist sd=HyTime
clause="A66000">

<propdef rcsnm=sosseqs appnm="sos sequences"
```

```
fullnm="alternative SOS sequences" nodelist subnode ac="sosseq"
sd=HyTime clause="A61003">

<classdef rcsnm=sosseq appnm="sos sequence" conprop=sequence sd=HyTime
clause="A61003">
<desc>
A storage object specification sequence.

<propdef rcsnm=sequence fullnm="sos sequence" nodelist subnode
ac="sos" sd=HyTime clause="A61003">

<classdef rcsnm=sos appnm="storage object spec"
fullnm="storage object specification" conprop=soi sd=HyTime
clause="A61100">

<propdef rcsnm=smname appnm="storage manager name" string
strnorm=general sd=HyTime clause="A61102">

<propdef rcsnm=records fullnm="record boundary indicator" enum
sd=HyTime clause="A65100">

<enumdef rcsnm=asis>
<enumdef rcsnm=rms fullnm="record management system">
<enumdef rcsnm=lf>
<enumdef rcsnm=cr>
<enumdef rcsnm=crlf>
<enumdef rcsnm=lfcr>
<enumdef rcsnm=find>

<propdef rcsnm=tracking appnm="record tracking" boolean sd=HyTime
clause="A65108">

<propdef rcsnm=bctf fullnm="bit combination transformation format"
string strnorm=general sd=HyTime clause="A65200">

<propdef rcsnm=extents appnm="occupied extents" nodelist subnode
ac="soextent" sd=HyTime clause="A65302">

<propdef rcsnm=zapeof appnm="zap eof" fullnm="zap end-of-file"
boolean sd=HyTime clause="A65304">

<propdef rcsnm=compress fullnm="compression information" string
strnorm=general sd=HyTime clause="A65307">

<propdef rcsnm=encrypt fullnm="encryption information" string
strnorm=general sd=HyTime clause="A65308">

<propdef rcsnm=seal fullnm="integrity information" string
strnorm=general sd=HyTime clause="A6530a">

<propdef rcsnm=soi appnm="storage object id"
fullnm="storage object identifier" string sd=HyTime clause="A61103">

<classdef rcsnm=soextent appnm="storage object extent" sd=HyTime
clause="A65302">
```

```
<propdef rcsnm=firstoct appnm="first octet" integer sd=HyTime
clause="A65302">

<propdef rcsnm=octcount appnm="octet count" integer sd=HyTime
clause="A65302">

<propdef rcsnm=lastoct appnm="last octet" integer sd=HyTime
clause="A65302">

</psmodule>

<!-- General Architecture classes and properties -->

<!-- Data Attributes For Elements abstract properties -->

<psmodule rcsnm=dafeabs fullnm="data attributes for elements abstract"
dependon=instabs>

<propdef cn=element rcsnm=notname string strnorm=general sd=HyTime
clause="A53001">
<when>
A notation was specified for the element.

<propdef cn=element rcsnm=notation node irefnode ac="notation"
sd=HyTime clause="A53001">
<when>
A notation was specified for the element.

<propdef cn=element rcsnm=dataatts appnm="data attributes" nmndlist
subnode ac="attasgn" acnmprop="name" sd=HyTime clause="A53003">
<when>
A notation was specified for the element and the DAFE facility of the
General Architecture was used.

</psmodule>

<psmodule rcsnm=gadcnabs
fullnm="general architecture common data attributes abstract"
dependon="prlgabs0 instabs">

<propdef cn=entity rcsnm=inclents appnm="included entities" nodelist
irefnode ac="entity" sd=HyTime clause="A57103">
<desc>
Entities included from notation data, specified using the "included"
common data attribute.

<propdef cn=entity rcsnm=altreps appnm="alt reps"
fullnm="alternative representations" nodelist irefnode ac="entity"
sd=HyTime clause="A57106">

<propdef cn=entity rcsnm=superdcn appnm="super dcn"
fullnm="notation derivation source" node irefnode ac="notation"
sd=HyTime clause="A57108">

</psmodule>
```

```
<!-- HyTime pseudo-element class -->

<psmodule rcsnm=pelement fullnm="pseudo-element abstract"
dependon="instabs">

<classdef rcsnm=pelement appnm="pseudo element"
fullnm="pseudo-element" sd=HyTime clause="71420">

<propdef rcsnm=content nodelist subnode
ac="datachar sdata extdata subdoc pi msignch ignrs ignre repos usemap
    uselink entstart entend ssep comdcl msstart msend ignmrkup"
sd=HyTime clause="71420">

</psmodule>
```

Annex B
(normative)

HyTime Property Set

This annex defines the HyTime property set.

The syntax and semantics of property definition documents are described in *A.4 Property Set Definition Requirements (PSDR)*.

B.1 Hyperdocuments and HyTime documents

A HyTime grove is the result of performing HyTime processing on each HyTime document within the bounded object set defined by a hub document. The root of a HyTime grove is a node of class **hyperdocument** (*hyperdoc*). The value of its **HyTime documents** (*hydocs*) property is list of **HyTime document** (*hydoc*) nodes, each of which corresponds with a HyTime document within the bounded object set. One of these documents is the hub document of the hyperdocument; the node corresponding to it is also the value of the **hub document** (*hubdoc*) property of the hyperdocument, and exhibits a true value for its **is hub document** (*hub*) property.

Each HyTime document node exhibits a value for its **effective SGML document** (*sgmldoc*) property that is the root of an SGML grove built from the HyTime document. This grove is identical to the grove that would be built by the SGML grove construction process from the HyTime document, with the exception that the grove plan used is that specified on the document element of the HyTime document, if specified. Otherwise the grove plan is the HyTime default SGML grove plan (see *7.1.4.2 HyTime Default SGML Grove Plan*).

B.2 HyTime Property Set

The HyTime property set is:

```
<!-- HyTime Property Set -->

<!DOCTYPE propset
   PUBLIC "ISO/IEC 10744:1997//DTD Property Set//EN"
[
<!NOTATION HyTime
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!NOTATION SGML
   PUBLIC "ISO 8879:1986//NOTATION
           Standard Generalized Markup Language (SGML)//EN"
>
]>

<propset nsd=HyTime>

<normdef rcsnm=general sd=SGML clause="d4506">
<desc>
Declared concrete syntax general namecase substitution of the relevant
HyTime document.

<normdef rcsnm=minlit sd=SGML clause="a1704">
<desc>
```

```
SGML minimum literal interpretation of the relevant HyTime document.

<classdef rcsnm=hyperdoc appnm="hyperdocument" conprop=hydocs
clause="62400 65000">
<desc>
The hyperdocument is the root of the HyTime grove, which is the result
of performing HyTime processing on the documents comprising the bounded
object set of the hyperdocument.

<!-- Bounded object set -->

<psmodule rcsnm=bos fullnm="bounded object set">

<propdef cn=hyperdoc rcsnm=bos fullnm="bounded object set" nmndlist
subnode ac="bosmem" acnmprop="memroot" clause="62400 65000">
<desc>
The list of groves comprising the bounded object set of the
hyperdocument.

<classdef rcsnm=bosmem appnm="bos member"
fullnm="bounded object set member" clause="62400 65000">
<desc>
A member of the hyperdocument's bounded object set.

<propdef rcsnm=inbos appnm="in hytime bos" boolean clause="65000 62400">
<desc>
Whether or not the BOS member is part of the HyTime BOS.

<propdef rcsnm=memroot fullnm="bos member root" node urefnode clause="65000 62400">
<desc>
The root of the BOS member's grove.

</psmodule>

<!-- Enabling Architecture Defined Semantics -->

<psmodule rcsnm=arcsem fullnm="enabling architecture defined semantics">

<propdef cn=hyperdoc rcsnm=arcsem appnm="arc semantics"
fullnm="enabling architecture defined semantics" nmndlist subnode
ac="arcsem" acnmprop="arcpubid"
clause="67300 82000 A31000 A34100">
<desc>
The list of enabling architectures used to define HyTime-significant
semantics within the hyperdocument.
<note>
Enabling architectures that are not identified using a public
identifier are not included within the list.  The scope of
HyTime-significant semantics defined within such an architecture is
restricted to the HyTime document that conforms to that architecture.

<classdef rcsnm=arcsem appnm="arc semantics"
fullnm="enabling architecture defined semantics"
clause="67300 82000 A31000 A34100">
<desc>
A collection of HyTime-significant enabling architecture-defined
```

```
semantics, such as activity and hyperlink types.

<propdef rcsnm=arcpubid appnm="arc public id"
fullnm="enabling architecture public identifier" string
strnorm=minlit clause="A34100">

</psmodule>

<!-- HyTime document -->

<psmodule rcsnm=hydoc fullnm="HyTime document">

<propdef cn=hyperdoc rcsnm=hubdoc appnm="hub document" node irefnode
ac="hydoc" clause="64000 65000">
<desc>
The HyTime document from which the bounded object set of the
hyperdocument was determined.

<propdef cn=hyperdoc rcsnm=hydocs appnm="hytime documents" nmndlist
subnode ac="hydoc" acnmprop="sgmldoc" clause="64000 65000">
<desc>
The list of HyTime documents within the bounded object set of the
hyperdocument.

<classdef rcsnm=hydoc appnm="hytime document" conprop=content
clause="64000 65000">
<desc>
A HyTime document included within the bounded object set of the
hyperdocument.

<propdef rcsnm=hub appnm="is hub document" boolean clause="65000">
<desc>
True if the HyTime document is the hub of the hyperdocument, otherwise
false.

<propdef rcsnm=content nmndlist subnode ac="desctab actrule locaddr
hylink fcs evsched rendrule modrule modpatch wand wandrule wndpatch
prorule proseq baton batrule batonseq projectr" acnmprop="def def def
def def def def def def def def def def def def def def def">

</psmodule>

<!-- Effective SGML document -->

<psmodule rcsnm=sgmldoc fullnm="effective SGML document"
dependon="hydoc">

<propdef cn=hydoc rcsnm=sgmldoc appnm="sgml document"
fullnm="effective SGML document" node urefnode clause="71430">
<desc>
The effective SGML grove of the document.

</psmodule>

<!-- Value Reference -->
```

```
<psmodule rcsnm=valueref fullnm="value reference">

<propdef cn=hyperdoc rcsnm=valrefs appnm="value refs"
fullnm="value references" nmndlist subnode ac="valueref"
acnmprop="valueof" clause="67100">

<classdef rcsnm=valueref appnm="value ref"
fullnm="value reference" clause="67100">

<propdef rcsnm=valueof appnm="value of" node urefnode clause="67100">
<desc>
The element or attribute assignment that is deemed to have the
referenced value.

<propdef rcsnm=value nodelist urefnode clause="67100">

</psmodule>

<!-- Description Table -->

<psmodule rcsnm=desctab fullnm="description table">

<propdef cn=valueref rcsnm=descent appnm="desc table entry"
fullnm="description table entry" node irefnode ac="descent" clause="67200">
<when>
The value reference was made through a description table.

<classdef rcsnm=desctab appnm="desc table"
fullnm="description table" conprop=entries clause="67220">

<propdef rcsnm=def appnm="definition" node urefnode clause="67220 71430">
<desc>
The element that defined the description table.

<propdef rcsnm=entries nmndlist subnode ac="descent"
acnmprop="desctxt" clause="67220">

<classdef rcsnm=descent appnm="desc table entry"
fullnm="description table entry" clause="67220">

<propdef rcsnm=desctxt appnm="desc text"
fullnm="descriptive text" string strnorm=minlit clause="67230">

<propdef rcsnm=descdef appnm="desc text def" node urefnode clause="67240 71430">
<desc>
The descdef element in the effective SGML grove of the HyTime document
associated with the descriptive text.

</psmodule>

<!-- Activity policy association rule -->

<psmodule rcsnm=actrule fullnm="activity policy association rule">

<classdef rcsnm=actrule appnm="activity policy rule"
fullnm="activity policy association rule" conprop=actpruls
```

```
clause="67308">

<propdef rcsnm=def appnm="definition" node urefnode
clause="71430 67308">
<desc>
The element that defined the activity policy association rule.

<propdef rcsnm=governed nodelist urefnode clause="67300">
<desc>
The nodes governed by the associated activity policies.

<propdef rcsnm=associat appnm="assoc or dissoc"
fullnm="associate or dissociate" enum clause="67300">
<enumdef rcsnm=assoc fullnm="associate">
<enumdef rcsnm=dissoc fullnm="dissociate">

<propdef rcsnm=nullify nodelist irefnode ac="actrule"
clause="67300">

<propdef rcsnm=actpruls appnm="activity type rules"
fullnm="activity type specific activity policy association rule portions"
nmndlist subnode ac="acttprul" acnmprop="typename"
clause="67300">

<classdef rcsnm=acttprul appnm="activity type rule"
fullnm="activity type specific activity policy association rule portion"
clause="67300">

<propdef rcsnm=typename appnm="activity type name" string strnorm=general
clause="67300">
<desc>
The name of the activity type.

<propdef rcsnm=policies fullnm="activity policies" nodelist urefnode
clause="67300">
<desc>
The activity policies governing the associated node for the given
activity type.

</psmodule>

<!-- Activity type -->

<psmodule rcsnm=acttype fullnm="activity type" dependon="arcsem hydoc">

<propdef cn=hydoc rcsnm=acttypes appnm="activity types" nmndlist
subnode ac="acttype" acnmprop="name" clause="67300 A31000">
<desc>
The activity types defined by the encompassing architecture of the
document.

<propdef cn=arcsem rcsnm=acttypes appnm="activity types" nmndlist
subnode ac="acttype" acnmprop="name" clause="67300 A31000">
<desc>
The activity types defined by the given enabling architecture.
```

```
<propdef cn=acttprul rcsnm=acttype appnm="activity type" node irefnode
ac="acttype" clause="67300">

<classdef rcsnm=acttype appnm="activity type" clause="67300">

<propdef rcsnm=name string strnorm=general clause="67300">

</psmodule>

<!-- Activity policy association -->

<psmodule rcsnm=actassoc fullnm="activity policy association"
dependon="acttype">

<propdef cn=hyperdoc rcsnm=actasscs appnm="activity policy assocs"
fullnm="activity policy associations" nmndlist subnode ac="actassoc"
acnmprop="governed" clause="67300">

<classdef rcsnm=actassoc appnm="activity policy assoc"
fullnm="activity policy associations" conprop=actpassc
clause="67300">

<propdef rcsnm=governed node urefnode clause="67300">
<desc>
The node governed by the associated activity policies.

<propdef rcsnm=actrules appnm="activity policy rules" nodelist
irefnode ac="actrule">
<desc>
The activity policy rules that apply to the governed node.

<propdef rcsnm=actpassc appnm="activity type assocs"
fullnm="activity type specific activity policy associations portion"
nmndlist subnode ac="actpassc" acnmprop="acttype"
clause="67300">

<classdef rcsnm=actpassc appnm="activity type assocs"
fullnm="activity type specific activity policy associations portion"
clause="67300">

<propdef rcsnm=acttype appnm="activity type" node irefnode ac="acttype"
clause="67300">

<propdef rcsnm=policies fullnm="activity policies" nodelist urefnode
clause="67300">
<desc>
The activity policies governing the associated node for the given
activity type.

</psmodule>

<!-- Location address targets -->

<psmodule rcsnm=loctarg fullnm="location address targets">

<propdef cn=hyperdoc rcsnm=reftargs appnm="reference targets" nmndlist
```

```
subnode ac="reftargs" acnmprop="ref" clause="71003">
<desc>
The resolved targets of references.

<classdef rcsnm=reftargs appnm="reference targets"
clause="71003">
<desc>
The result of resolving the targets of a reference.

<propdef rcsnm=ref appnm="reference" node urefnode
clause="71003">
<desc>
The node making the reference.

<propdef rcsnm=targets nodelist urefnode clause="71003">
<desc>
The resolved targets of the reference.

</psmodule>

<!-- Location address path -->

<psmodule rcsnm=locpath fullnm="location address path"
dependon="hydoc">

<propdef cn=reftargs rcsnm=locpath appnm="location path"
fullnm="location address path" node irefnode ac="locaddr"
clause="71300">
<desc>
The first step of the location path used to resolve the targets of the
reference.
<when>
A location path was used to resolve the targets of the reference.

<classdef rcsnm=locaddr appnm="location address"
clause="71000">
<desc>
An abstract HyTime location address.

<propdef rcsnm=def appnm="definition" node urefnode
clause="71430">
<desc>
The element or attribute assignment that defined the location address.

<propdef rcsnm=locsrc appnm="location source" nodelist urefnode
clause="72000 73000">

<propdef rcsnm=selector appnm="location selector" nodelist urefnode
clause="70000">
<desc>
That portion of the location address definition used to select the
located nodes from the location source.
<note>
For instance, the selector of each node located by a name space
location address is the corresponding name in the nmsploc element's
content.
```

```
<propdef rcsnm=locnodes appnm="located nodes" nodelist urefnode
clause="70000">
<desc>
The nodes selected from the location source by the location selector.
<note>
If the value exhibited for the located nodes property by a location
address contains more than one node, the location address is a
multiple location address.

</psmodule>

<!-- Hyperlink -->

<psmodule rcsnm=hylink fullnm="hyperlink" dependon="hydoc">

<propdef cn=hyperdoc rcsnm=anchobjs appnm="anchored objects" nmndlist
subnode ac="anchobj" acnmprop="object" clause="80000">
<desc>
A mapping of objects to anchors.

<classdef rcsnm=anchobj appnm="anchored object"
clause="81100">
<desc>
The hyperlink anchors for which a given node is the anchored object.

<propdef rcsnm=object node urefnode clause="81100">
<desc>
The anchored object.

<propdef rcsnm=anchors nodelist irefnode ac="anchor"
clause="81100">
<desc>
The anchors that anchor the object.

<classdef rcsnm=hylink appnm="hyperlink"
clause="81100 82000">

<propdef rcsnm=def appnm="definition" node urefnode
clause="81106">
<desc>
The element that defined the hyperlink.

<propdef rcsnm=typename appnm="hyperlink type name" string
strnorm=general clause="81107">

<propdef rcsnm=anchors nmndlist subnode ac="anchor listanch"
acnmprop="rolename rolename" clause="81100 82100">

<classdef rcsnm=listanch appnm="list anchor"
clause="81100 82100">

<propdef rcsnm=rolename appnm="anchor role name" string
strnorm=general clause="81100 82105">

<propdef rcsnm=members nodelist subnode ac="anchor"
```

```
clause="82100">

<classdef rcsnm=anchor clause="81100">

<propdef rcsnm=rolename appnm="anchor role name" string
strnorm=general clause="81100 82105">

<propdef rcsnm=object fullnm="anchored object" node urefnode
clause="81100">

<propdef rcsnm=return fullnm="return traversal" boolean
clause="81300">

<propdef rcsnm=depart appnm="departure" fullnm="hyperlink departure"
boolean clause="81300">

<propdef rcsnm=intarrt appnm="int arrival trav"
fullnm="internal arrival traversals" nodelist irefnode ac="anchor"
clause="81200 81300">

<propdef rcsnm=extarrt appnm="ext arrival trav"
fullnm="external arrival traversals" nodelist irefnode ac="anchor"
clause="81200 81300">

<propdef rcsnm=nextleft appnm="next left"
fullnm="next member to the left" node irefnode ac="anchor"
clause="81200 81300">
<when>
The anchor is a member of a list anchor and left member traversal is
allowed, unless the anchor is the leftmost member of the list and
wrapping member traversal is not allowed.

<propdef rcsnm=nxtright appnm="next right"
fullnm="next member to the right" node irefnode ac="anchor"
clause="81200 81300">
<when>
The anchor is a member of a list anchor and right member traversal is
allowed, unless the anchor is the rightmost member of the list and
wrapping member traversal is not allowed.

</psmodule>

<!-- Hyperlink type -->

<psmodule rcsnm=hylinktp fullnm="hyperlink type" dependon="arcsem hydoc">

<propdef cn=hydoc rcsnm=hylktps appnm="hyperlink types" nmndlist
subnode ac="hylinktp" acnmprop="name" clause="81100 A31000">
<desc>
The hyperlink types defined by the encompassing architecture of the
document.

<propdef cn=arcsem rcsnm=hylktps appnm="hyperlink types" nmndlist
subnode ac="hylinktp" acnmprop="name" clause="81100 A31000">
<desc>
The hyperlink types defined by the given enabling architecture.
```

```
<propdef cn=hylink rcsnm=hylinktp appnm="hyperlink type" node irefnode
ac="hylinktp" clause="81100">

<propdef cn=listanch rcsnm=anchrole appnm="anchor role" node irefnode
ac="anchrole" clause="81100 82100">

<propdef cn=anchor rcsnm=anchrole appnm="anchor role" node irefnode
ac="anchrole" clause="81100 82100">

<classdef rcsnm=hylinktp appnm="hyperlink type"
clause="81100 82100">

<propdef rcsnm=name string strnorm=general clause="81100 82100">

<propdef rcsnm=roles nmndlist subnode ac="anchrole" acnmprop="name"
clause="81100 82105">

<classdef rcsnm=anchrole appnm="anchor role"
clause="81100 82105">

<propdef rcsnm=name string strnorm=general clause="81100 82105">

<propdef rcsnm=list boolean clause="82100">
<desc>
Whether or not the role is filled by a list anchor.

</psmodule>

<!-- Finite coordinate space -->

<psmodule rcsnm=fcs fullnm="finite coordinate space" dependon="hydoc">

<classdef rcsnm=fcs fullnm="finite coordinate space" clause="93000">

<propdef rcsnm=def appnm="definition" node urefnode clause="93000 71430">
<desc>
The element that defined the finite coordinate space.

<propdef rcsnm=axes nmndlist subnode ac="axis" acnmprop="name"
clause="93003">

<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode
ac="dim" acnmprop="axisnm" clause="98203">

<classdef rcsnm=axis clause="93000">

<propdef rcsnm=name string strnorm=general clause="93005">

<propdef rcsnm=smu fullnm="standard measurement unit" node urefnode
clause="93003 92100">
<desc>
The standard measurement unit for the axis (a notation node in an SGML
grove).

<propdef rcsnm=smumdurn appnm="smu to mdu ratio numerator"
```

```
integer clause="9300c">

<propdef rcsnm=smumdurd appnm="smu to mdu ratio denominator"
integer clause="9300c">

<propdef rcsnm=qcount appnm="quantum count" integer clause="9300d 68004">

<classdef rcsnm=dim appnm="dimension" clause="68004 68200">

<propdef rcsnm=axisnm appnm="axis name" string strnorm=general
clause="93005">

<propdef rcsnm=axis node irefnode ac="axis" clause="93000">

<propdef rcsnm=firstq appnm="first quantum" integer clause="68004">

<propdef rcsnm=qcount appnm="quantum count" integer clause="68004">

<propdef rcsnm=lastq appnm="last quantum" integer clause="68004">

<classdef rcsnm=extent clause="94400 95100">

<propdef rcsnm=dimens appnm=dimensions nmndlist subnode ac="dim"
acnmprop="axisnm" clause="94000 68200">

</psmodule>

<!-- Finite coordinate space type -->

<psmodule rcsnm=fcstype fullnm="finite coordinate space type"
dependon="arcsem hydoc">

<propdef cn=hydoc rcsnm=fcstypes appnm="fcs types"
fullnm="finite coordinate space types" nmndlist subnode ac="fcstype"
acnmprop="name" clause="93000 A31000">
<desc>
The FCS types defined by the encompassing architecture of the
document.

<propdef cn=arcsem rcsnm=fcstypes appnm="fcs types"
fullnm="finite coordinate space types" nmndlist subnode ac="fcstype"
acnmprop="name" clause="93000 A31000">
<desc>
The FCS types defined by the given enabling architecture.

<propdef cn=fcs rcsnm=fcstype appnm="fcs type"
fullnm="finite coordinate space type" node irefnode ac="fcstype"
clause="93000">

<propdef cn=axis rcsnm=axistype appnm="axis type"
fullnm="finite coordinate space axis type" node irefnode
ac="axistype" clause="93000">

<classdef rcsnm=fcstype appnm="fcs type"
fullnm="finite coordinate space type" clause="93000">
```

```
<propdef rcsnm=name string strnorm=general clause="93000">

<propdef rcsnm=axes nmndlist subnode ac="axistype" acnmprop="name"
clause="93000">

<classdef rcsnm=axistype appnm="axis type"
fullnm="finite coordinate space axis type" clause="93000">

<propdef rcsnm=name string strnorm=general clause="93000">

<propdef rcsnm=smu fullnm="standard measurement unit" node urefnode
clause="93003 92100">
<desc>
The standard measurement unit for the axis (a notation node in an SGML
grove).

</psmodule>

<!-- Axis calibration -->

<psmodule rcsnm=axiscal fullnm="axis calibration">

<propdef cn=axis rcsnm=calquant appnm="calibration quantum" integer
clause="93103">
<desc>
The number of the quantum before or after which the axis is calibrated.
<when>
A calibration was specified for the axis.

<propdef cn=axis rcsnm=calpoint appnm="calibration point" enum
clause="93103">
<desc>
The position of the calibration point relative to the calibration
quantum.
<when>
A calibration was specified for the axis.

<enumdef rcsnm=start>
<enumdef rcsnm=end>

<propdef cn=axis rcsnm=calibrat appnm="calibration" string clause="93100">
<when>
A calibration was specified for the axis.

<propdef cn=axis rcsnm=calnot appnm="calibration notation" node urefnode
clause="93104">
<desc>
The notation node (in an SGML grove) for the notation used to specify
the calibration.
<when>
A calibration was specified for the axis.

</psmodule>

<!-- Event Schedule -->
```

```
<psmodule rcsnm=evsched fullnm="scheduling" dependon="hydoc">

<propdef cn=hyperdoc rcsnm=schdobjs appnm="scheduled objects" nmndlist
subnode ac="schdobj" acnmprop="object" clause="95100 96000">

<propdef cn=fcs rcsnm=evscheds appnm="event schedules" nodelist irefnode
ac="evsched" clause="93004">

<classdef rcsnm=schdobj appnm="scheduled object" clause="95100">

<propdef rcsnm=object node urefnode clause="95104 96000">

<propdef rcsnm=events nodelist irefnode ac="event" clause="95100">

<classdef rcsnm=evsched appnm="event schedule" conprop=events
clause="95000">

<propdef rcsnm=def appnm="definition" node urefnode clause="95000 71430">
<desc>
The element that defined the event schedule.

<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode ac="dim"
acnmprop="axisnm" clause="98203">

<propdef rcsnm=events nodelist subnode ac="evgrp event"
clause="95100 95200">

<propdef rcsnm=sparse boolean clause="94105">
<desc>
Whether or not gaps may exist between events within the schedule.

<propdef rcsnm=overlap boolean clause="94106">
<desc>
Whether or not events within the schedule may overlap.

<classdef rcsnm=evgrp appnm="event group" conprop=events
clause="95200">

<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode ac="dim"
acnmprop="axisnm" clause="98203">

<propdef rcsnm=forced appnm="extent forced" boolean clause="94300">
<desc>
True if extents of contained events are derived from extent of group
as opposed to the occupied extent of the group being derived from the
extents of the contained events.

<propdef rcsnm=events nodelist subnode ac="evgrp event"
clause="95100 95200">

<classdef rcsnm=event clause="95100">

<propdef rcsnm=objects nodelist urefnode clause="95104 96000">

<propdef rcsnm=extents nodelist subnode ac="extent" clause="95102 94000">
```

```
</psmodule>

<!-- Modifier patch -->

<psmodule rcsnm=modpatch fullnm="modifier patch">

<classdef rcsnm=modpatch appnm="modifier patch" clause="a2400">

<propdef rcsnm=def appnm="definition" node urefnode clause="a2400 71430">
<desc>
The element that defined the modifier patch.
<when>
The modifier patch was not generated from a wand patch.

<propdef rcsnm=patch node subnode ac="modspec modseq modfork" clause="a2400">

<classdef rcsnm=modspec appnm="modifier spec"
fullnm="modifier specification" clause="a2400">

<propdef rcsnm=modifier fullnm="modifier or modifier patch"
node urefnode clause="a2400">

<classdef rcsnm=modseq appnm="modifier sequence" clause="a2404">

<propdef rcsnm=patch node subnode ac="modspec modseq modfork" clause="a2400">

<classdef rcsnm=modfork appnm="modifier fork" clause="a2404">
<desc>
A set of modifiers to be applied in parallel.

<propdef rcsnm=patch node subnode ac="modspec modseq modfork" clause="a2400">

</psmodule>

<!-- Wand -->

<psmodule rcsnm=wand dependon="fcs">

<propdef cn=hyperdoc rcsnm=schdmods appnm="scheduled modifiers" nmndlist
subnode ac="schedmod" acnmprop="modifier" clause="a2000">

<propdef cn=fcs rcsnm=wands nodelist irefnode ac="wand" clause="93004 a2320">

<classdef rcsnm=schedmod appnm="scheduled modifier" clause="a2300 a2330 a2340">

<propdef rcsnm=modifier node urefnode clause="a2100">

<propdef rcsnm=modscops appnm="modifier scopes" nodelist irefnode
ac="modscope" clause="a2330">

<classdef rcsnm=wand conprop=scopes clause="a2320">

<propdef rcsnm=def appnm="definition" node urefnode clause="a2320 71430">
<desc>
The element that defined the wand.
```

```
<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode ac="dim"
acnmprop="axisnm" clause="98203">

<propdef rcsnm=scopes appnm="modscopes" fullnm="modifier scopes"
nodelist subnode ac="modgrp modscope" clause="a2330 a2340">

<propdef rcsnm=sparse boolean clause="94105">
<desc>
Whether or not gaps may exist between modifier scopes within the
wand.

<propdef rcsnm=overlap boolean clause="94106">
<desc>
Whether or not modifier scopes within the wand may overlap.

<classdef rcsnm=modgrp appnm="modifier scope group" conprop=scopes clause="a2340">

<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode ac="dim"
acnmprop="axisnm" clause="98203">

<propdef rcsnm=forced appnm="extent forced" boolean clause="94300">
<desc>
True if extents of contained modifier scopes are derived from extent
of group as opposed to the occupied extent of the group being derived
from the extents of the contained modifier scopes.

<propdef rcsnm=scopes appnm="modscopes" fullnm="modifier scopes"
nodelist subnode ac="modgrp modscope" clause="a2330 a2340">

<classdef rcsnm=modscope appnm="modifier scope" clause="a2330">

<propdef rcsnm=modifier nodelist urefnode clause="a2332 a2100">

<propdef rcsnm=extents nodelist subnode ac="extent" clause="94200 94000">

</psmodule>

<!-- Wand patch -->

<psmodule rcsnm=wndpatch fullnm="wand patch">

<classdef rcsnm=wndpatch appnm="wand patch" clause="a2400">

<propdef rcsnm=def appnm="definition" node urefnode clause="a2400 71430">
<desc>
The element that defined the wand patch.

<propdef rcsnm=patch node subnode ac="wandspec wandseq wandfork" clause="a2400">

<classdef rcsnm=wandspec appnm="wand spec"
fullnm="wand specification" clause="a2400">

<propdef rcsnm=wand fullnm="wand or wand patch" node irefnode
ac="wand wndpatch" clause="a2400">

<classdef rcsnm=wandseq appnm="wand sequence" clause="a2404">
```

```
<propdef rcsnm=patch node subnode ac="wandspec wandseq wandfork" clause="a2400">

<classdef rcsnm=wandfork appnm="wand fork" clause="a2404">
<desc>
A set of wands to be applied in parallel.

<propdef rcsnm=patch node subnode ac="wandspec wandseq wandfork" clause="a2400">

</psmodule>

<!-- Object modification -->

<psmodule rcsnm=modify fullnm="object modification">

<propdef cn=event rcsnm=mods appnm="modifications" nodelist subnode
ac="mod" clause="a2000">

<propdef cn=modscope rcsnm=mods appnm="modifications" nodelist subnode
ac="mod" clause="a2000">

<classdef rcsnm=mod appnm="modification" clause="a2000">

<propdef rcsnm=extents appnm="selected extents" nodelist subnode
ac="selext" clause="a2300">
<desc>
The extents that are subject to the modification.

<propdef rcsnm=modifier fullnm="modifier or modifier patch"
node urefnode clause="a2100 a2400">
<desc>
The modifier node (in another grove) or modifier patch
node (in this grove) that describe the modification.

<propdef rcsnm=genmdpat appnm="gen modifier patch"
fullnm="generated modifier patch" node subnode ac="modpatch" clause="a2310 a2400">
<when>
The modification was applied by a wand rule.

<propdef rcsnm=rule fullnm="modifier or wand rule" node irefnode
ac="modrule wandrule" clause="a2200 a2310">
<desc>
The modifier or wand rule that applied the modification.

<propdef rcsnm=modscope appnm="modifier scope" node irefnode
ac="modscope" clause="a2330">
<desc>
The modifier scope that caused the event to be selected for
modification.
<when>
The modification was applied by a wand rule.

</psmodule>

<!-- Modifier rule -->
```

```
<psmodule rcsnm=modrule fullnm="modifier rule">

<classdef rcsnm=modrule appnm="modifier rule" clause="a2200">

<propdef rcsnm=def appnm="definition" node urefnode clause="a2200 71430">
<desc>
The element that defined the modifier rule.

<propdef rcsnm=events nodelist irefnode
ac="event evgrp evsched modscope modgrp wand" clause="a2200">

<propdef rcsnm=modifier fullnm="modifier or modifier patch"
node urefnode clause="a2200">
<desc>
The modifier node (in another grove) or modifier patch
node (in this grove) that describe the modification.

</psmodule>

<!-- Wand rule -->

<psmodule rcsnm=wandrule fullnm="wand rule">

<classdef rcsnm=wandrule appnm="wand rule" clause="a2310">

<propdef rcsnm=def appnm="definition" node urefnode clause="a2310 71430">
<desc>
The element that defined the wand rule.

<propdef rcsnm=events nodelist irefnode
ac="event evgrp evsched modscope modgrp wand" clause="a2310">

<propdef rcsnm=wand fullnm="wand or wand patch"
node irefnode ac="wand wndpatch" clause="a2310">

</psmodule>

<!-- Projector -->

<psmodule rcsnm=projectr fullnm="projector" dependon="hydoc">

<classdef rcsnm=projectr appnm="projector" clause="a3100">

<propdef rcsnm=def appnm="definition" node urefnode clause="a3100 71430">
<desc>
The element that defined the projector.

<propdef rcsnm=strict appnm="strictness" string clause="a3104">
<desc>
Strict projection if null, otherwise application dependent.
<when>
A strictness was specified for the projector.

</psmodule>

<!-- Projector sequence -->
```

```
<psmodule rcsnm=proseq fullnm="projector sequence">

<classdef rcsnm=proseq appnm="projector seq"
fullnm="projector sequence" clause="a3230">

<propdef rcsnm=def appnm="definition" node urefnode clause="a3230 71430">
<desc>
The element that defined the projector sequence.
<when>
The projector sequence was not generated from a baton sequence.

<propdef rcsnm=sequence nodelist irefnode ac="projectr proseq" clause="a3230">

</psmodule>

<!-- Baton -->

<psmodule rcsnm=baton dependon="fcs">

<propdef cn=hyperdoc rcsnm=schdprjs appnm="scheduled projectors" nmndlist
subnode ac="schdproj" acnmprop="projectr" clause="a3300">

<propdef cn=fcs rcsnm=batons nodelist irefnode ac="baton" clause="a3320">

<classdef rcsnm=schdproj appnm="scheduled projector" clause="a3332">

<propdef rcsnm=projectr appnm="projector" node irefnode ac="projectr"
clause="a3100">

<propdef rcsnm=proscops appnm="projector scopes" nodelist irefnode
ac="proscope" clause="a3330">

<classdef rcsnm=baton conprop=scopes clause="a3320">

<propdef rcsnm=def appnm="definition" node urefnode clause="a3320 71430">
<desc>
The element that defined the baton.

<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode ac="dim"
acnmprop="axisnm" clause="98203">

<propdef rcsnm=scopes appnm="proscopes" fullnm="projector scopes"
nodelist subnode ac="progrp proscope" clause="a3330">

<propdef rcsnm=sparse boolean clause="94105">
<desc>
Whether or not gaps are allowed between projector scopes within the
baton.

<propdef rcsnm=overlap boolean clause="94106">
<desc>
Whether or not projector scopes within the baton are allowed to
overlap.

<classdef rcsnm=progrp appnm="projector scope group" conprop=scopes clause="a3340">
```

```
<propdef rcsnm=occext appnm="occupied extent" nmndlist subnode ac="dim"
acnmprop="axisnm" clause="98203">

<propdef rcsnm=forced appnm="extent forced" boolean clause="94300">
<desc>
True if extents of contained projector scopes are derived from extent
of group as opposed to the occupied extent of the group being derived
from the extents of the contained projector scopes.

<propdef rcsnm=scopes appnm="proscopes" fullnm="projection scopes"
nodelist subnode ac="progrp proscope" clause="a3330 a3340">

<classdef rcsnm=proscope appnm="projector scope" clause="a3330">

<propdef rcsnm=projectr appnm="projectors" nodelist irefnode
ac="projectr" clause="a3330 a3100">

<propdef rcsnm=extents nodelist subnode ac="extent" clause="94200 94000">

</psmodule>

<!-- Baton sequence -->

<psmodule rcsnm=batonseq fullnm="baton sequence">

<classdef rcsnm=batonseq appnm="baton seq"
fullnm="baton sequence" clause="a3350">

<propdef rcsnm=def appnm="definition" node urefnode clause="a3350 71430">
<desc>
The element that defined the baton sequence.

<propdef rcsnm=sequence nodelist irefnode ac="baton batonseq" clause="a3350">

</psmodule>

<!-- Projection -->

<psmodule rcsnm=project fullnm="projection">

<propdef cn=event rcsnm=pros appnm="projections" nodelist subnode
ac="pro" clause="a3000">

<propdef cn=modscope rcsnm=pros appnm="projections" nodelist subnode
ac="pro" clause="a3000">

<propdef cn=proscope rcsnm=pros appnm="projections" nodelist subnode
ac="pro" clause="a3000">

<propdef cn=event rcsnm=srcproj appnm="source projection" node
irefnode ac="pro" clause="a3000">
<desc>
The projection that resulted in the event.
<when>
The event is not indigenous to the schedule.
```

```
<propdef cn=modscope rcsnm=srcproj appnm="source projection" node
irefnode ac="pro" clause="a3000">
<desc>
The projection that resulted in the modifier scope.
<when>
The modifier scope is not indigenous to the wand.

<propdef cn=proscope rcsnm=srcproj appnm="source projection" node
irefnode ac="pro" clause="a3000">
<desc>
The projection that resulted in the projector scope.
<when>
The projector scope is not indigenous to the baton.

<classdef rcsnm=pro appnm="projection" clause="a3000 a3100 a3200 a3300">

<propdef rcsnm=extents appnm="selected extents" nodelist subnode
ac="selext" clause="a3300">
<desc>
The extents that are subject to the projection.

<propdef rcsnm=projectr appnm="projector"
fullnm="projector or projector sequence" node irefnode
ac="projectr proseq" clause="a3100 a3230">

<propdef rcsnm=genprosq appnm="gen projector seq"
fullnm="generated projector sequence" node subnode ac="proseq" clause="a3310">
<when>
The projection was applied by a baton rule.

<propdef rcsnm=result node irefnode ac="event modscope proscope" clause="a3000">
<desc>
The event, modscope, or proscope in the target schedule that
represents the result of the projection.

<propdef rcsnm=rule fullnm="projector or baton rule" node irefnode
ac="prorule batrule" clause="a3220 a3310">
<desc>
The projector or baton rule that applied the projection.

<propdef rcsnm=proscope appnm="projector scope" node irefnode
ac="proscope" clause="a3330">
<desc>
The projector scope that caused the event to be selected for
projection.
<when>
The projection was applied by a baton rule.

</psmodule>

<!-- Projector rule -->

<psmodule rcsnm=prorule fullnm="projector rule">

<classdef rcsnm=prorule appnm="projector rule" clause="a3220">
```

```
<propdef rcsnm=def appnm="definition" node urefnode clause="a3220 71430">
<desc>
The element that defined the projector rule.

<propdef rcsnm=events nodelist irefnode
ac="event evgrp evsched modscope modgrp wand proscope progrp baton" clause="a3220">

<propdef rcsnm=projectr appnm="projector"
fullnm="projector or projector sequence" node irefnode
ac="projectr proseq" clause="a3220">

<propdef rcsnm=targschd appnm="target schedules" nodelist irefnode
ac="evsched wand baton" clause="a3220">

</psmodule>

<!-- Baton rule -->

<psmodule rcsnm=batrule fullnm="baton rule">

<classdef rcsnm=batrule appnm="baton rule" clause="a3310">

<propdef rcsnm=def appnm="definition" node urefnode clause="a3310 71430">
<desc>
The element that defined the baton rule.

<propdef rcsnm=events nodelist irefnode
ac="event evgrp evsched modscope modgrp wand proscope progrp baton" clause="a3310">

<propdef rcsnm=baton fullnm="baton or baton sequence"
node irefnode ac="baton batonseq" clause="a3310">

<propdef rcsnm=targschd appnm="target schedules" nodelist irefnode
ac="evsched wand baton" clause="a3310">

</psmodule>

<!-- Rendition rule -->

<psmodule rcsnm=rendrule fullnm="rendition rule">

<classdef rcsnm=rendrule fullnm="rendition rule" clause="a4000">

<propdef rcsnm=def appnm="definition" node urefnode clause="a4000 71430">
<desc>
The element that defined the rendition rule.

<propdef rcsnm=rules nodelist irefnode ac="modrule wandrule prorule
batrule rendrule" clause="a4000">
<desc>
The rules that constitute the rendition.

</psmodule>

<!-- Rendition -->
```

```
<psmodule rcsnm=rend fullnm="rendition" dependon="rendrule">

<propdef cn=event rcsnm=rends appnm="renditions" nmndlist subnode
ac="rend" acnmprop="rendrule" clause="a0000">
<desc>
The list of renditions defined for the event.

<propdef cn=modscope rcsnm=rends appnm="renditions" nmndlist subnode
ac="rend" acnmprop="rendrule" clause="a0000">
<desc>
The list of renditions defined for the modifier scope.

<propdef cn=proscope rcsnm=rends appnm="renditions" nmndlist subnode
ac="rend" acnmprop="rendrule" clause="a0000">
<desc>
The list of renditions defined for the projector scope.

<classdef rcsnm=rend appnm="rendition" clause="a0000">

<propdef rcsnm=rendrule appnm="rendition rule" node irefnode
ac="rendrule" clause="a4000">

<propdef rcsnm=mods appnm="modifications" nodelist irefnode
ac="mod" clause="a0000">
<when>
The rendition is of an event or modifier scope.

<propdef rcsnm=pros appnm="projections" nodelist irefnode
ac="pro" clause="a0000">

</psmodule>

<!-- Selected extent -->

<psmodule rcsnm=selext fullnm="selected extent">

<classdef rcsnm=selext appnm="selected extent" clause="a1000 94000">
<desc>
An extent of an event selected for rendition.

<propdef rcsnm=seldims appnm="selected dimensions" nmndlist subnode
ac="seldim" acnmprop="axisnm" clause="a1000 94000">

<classdef rcsnm=seldim appnm="selected dimension" clause="a1000 94000 68200">
<desc>
A dimension of an event selected for rendition.

<propdef rcsnm=axisnm appnm="axis name" string strnorm=general clause="93005">

<propdef rcsnm=axis node irefnode ac="axis" clause="93000">

<propdef rcsnm=firstq appnm="first quantum" integer clause="68004">

<propdef rcsnm=qcount appnm="quantum count" integer clause="68004">
```

```
<propdef rcsnm=lastq appnm="last quantum" integer clause="68004">

<propdef rcsnm=whole boolean clause="a1106">
<desc>
Whether or not the selected dimension was deemed to be the whole
dimension by the relevant "portion affected" attribute.

</psmodule>

<!-- Pulse maps -->

<psmodule rcsnm=pulsemap fullnm="pulse maps">

<propdef cn=evsched rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=evgrp rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=event rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=wand rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=modgrp rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=modscope rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=baton rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=progrp rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

<propdef cn=proscope rcsnm=pulsmaps appnm="pulse maps" nodelist
irefnode ac="evsched wand baton" clause="97000">

</psmodule>
```

Annex C
(normative)

Architectural Meta-Declarations

This annex contains the complete architectural meta-declarations for the architectures defined in this standard. Each meta-DTD reflects the exact architectural form declarations given in the text of the standard integrated with the other declarations necessary for a complete and functioning architectural meta-declaration set.

C.1 HyTime Lexical Types

This clause defines lexical types used in the definition of HyTime architectural forms, as well as generally useful lexical types. These lexical type declarations conform to the requirements defined in *A.2 Lexical Type Definition Requirements (LTDR)*.

```
<!--
This file is identified by the following public identifier:

"ISO/IEC 10744:1997//NONSGML LEXTYPES HyTime Lexical Types//EN"
-->

<!ENTITY % SGMLlex
   PUBLIC "ISO/IEC 10744//NONSGML LEXTYPES SGML Lexical Types//EN"
>
%SGMLlex;

              <!-- POSIX Regular Expression Notation -->

<!NOTATION REGEX
   PUBLIC "ISO/IEC 9945-2:1997//NOTATION
           POSIX Regular Expression Notation//EN">
<!ATTLIST #NOTATION REGEX
   case     (case|icase) case
>

                    <!-- HyTime Lexical Types -->

<!LEXTYPE
   str            -- Unnormalized string --

   "[char]*"
   HyLex [unorm]
>
<!LEXTYPE
   norm           -- Normalized text --

   "s*,[(#NOT s)*],(s+,[(#NOT s)*])*,s*"
   HyLex [unorm]
>
<!LEXTYPE
   word           -- Word --

   "nmchar+"
   HyLex [unorm]
```

```
>
<!LEXTYPE
   words          -- Words --

   "word+"
   HyLex
>
<!LEXTYPE
   line           -- Line --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXTYPE Line//EN"
>

<!LEXTYPE
   ATTORCON       -- Attribute name or content --

   'ATTNAME|"#CONTENT"'
   HyLex [unorm]
>
<!LEXCON
   SMU            -- Standard measurement unit --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXCON
           Standard Measurement Unit//EN"
>
<!LEXTYPE
   SMU            -- Standard measurement unit --

   #CHECK SMU

   "NOTATION"
   HyLex [unorm]
>
<!LEXCON
   AXISNM         -- Axis name --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXCON
           Axis name//EN"
>
<!LEXTYPE
   AXISNM         -- Axis name --

   #CHECK AXISNM

   "NAME"
   HyLex [unorm]
>
<!LEXCON
   granule        -- Measurement granule name --

   SPEC
   PUBLIC "ISO/IEC 10744:1997//NOTATION LEXCON
           Measurement Granule Name//EN"
```

```
>
<!LEXTYPE
   granule        -- Measurement granule name --

   #CHECK granule

   "nmstrt,nmchar*"
   HyLex [unorm]
>
<!LEXTYPE
   ratio          -- Measurement granule name --

   "unzi,unzi"
   HyLex
>
<!LEXTYPE
   unzi           -- Unsigned non-zero integer --

   "[1-9][0-9]*"
   REGEX
>
<!LEXTYPE
   snzi           -- Signed non-zero integer --

   "[+-]?[1-9][0-9]*"
   REGEX
>
<!LEXTYPE
   marker         -- HyTime Axis marker --

   "snzi"
   HyLex [unorm]
>
```

Taken together, the lexical type declarations defined in this clause and the lexical types defined for SGML form the HyTime Lexical Types definition document. This document begins with the following declarations:

The HyTime Lexical Type definition document is referenced from the HyTime meta-DTD using a processing instruction as defined in *A.2 Lexical Type Definition Requirements (LTDR)*.

C.1.1 Calendar-Related Lexical Types

```
<!LEXTYPE
   JULDATE        -- Julian date --

   '"JD" #ORDER SGMLCASE,"-"?,NUMBER'
   HyLex [unorm]
>
<!LEXTYPE
   UTC            -- UTC date and time --

   "UTCdate,UTCtime"
   HyLex
>
<!LEXTYPE
```

```
   fulldate       -- Full date --

   "UTCdate, ERA?"
   HyLex
>
<!LEXTYPE
   ERA            -- Era --

   '"BC"|"BCE"|"BP"|"AD"|"CE"'
   HyLex
>
<!LEXTYPE
   UTCdate        -- UTC date in era --

   'year,"-"?,month,"-"?,day'
   HyLex [unorm]
>
<!LEXTYPE
   monthday       -- Month and day in year --

   'month,"-"?,day'
   HyLex [unorm]
>
<!LEXTYPE
   year           -- Year in era --

   "[0-9][0-9][0-9][0-9]"
   REGEX
>
<!LEXTYPE
   month          -- Month in year --

   "0[1-9]|1[0-2]"
   REGEX
>
<!LEXTYPE
   day            -- Day in month --

   "[0-2][1-9]|[3][0-1]"
   REGEX
>
<!LEXTYPE
   UTCtime        -- UTC time of day --

   'hrminsec,(".",Digit+)?'
   HyLex [unorm]
>
<!LEXTYPE
   hrminsec       -- Time of day in hh:mm:ss format --

   'hour,":"?,minute,":"?,second'
   HyLex [unorm]
>
<!LEXTYPE
   hrmin          -- Time of day in hh:mm format --
   'hour,":"?,minute'
   HyLex [unorm]
>
<!LEXTYPE
   hour           -- Hour of day --

   "[0-1][0-9]|[2][0-3]"
   REGEX
>
<!LEXTYPE
   minute         -- Minute of hour --

   "[0-5][0-9]"
   REGEX
>
<!LEXTYPE
   second         -- Second of minute --

   "[0-5][0-9]|60"
   REGEX
>
<!LEXTYPE
   timeoff        -- Timezone offset boundary --

   "offset,(monthday,hour)?"
   HyLex
>
<!LEXTYPE
   offset         -- Time offset --

   '("+"|"-")?,hrmin'
   HyLex [unorm]
>
```

C.2 HyTime Meta-Declarations

The HyTime meta-declaration set is:

```
<!AFDR "ISO/IEC 10744:1997">

<!--
ISO/IEC 10744:1997//DTD AFDR Meta-DTD
Hypermedia/Time-based Structuring Language
(HyTime)//EN
-->

<!--

Option Summary:

[General Architecture]

altreps   Alternative Representation Facility
dafe      Data Attributes for Elements Facility
dvlist    Default Value List Facility
```

HyLex	HyTime Lexical Model Notation
HyOrd	HyTime Lexicographic Ordering Notation
included	Included Entities Facility Facility
ireftype	Immediate ID Reference Type Facility
lextype	Lexical Typing Facility
opacity	Element Opacity Facility
REGEX	POSIX Regular Expression Notation
superdcn	Notation Derivation Source Facility

[Base Module]

activity	Activity Policy Association Facility
actypes	User-defined Activity Type Facility
bos	HyTime Bounded Object Set Specification Facility
bosspec	HyTime BOS Exception Specification Facility
conloc	Content Location Facility
desctxt	Descriptive Text Facility
dimspec	Dimension Specification Facility
HyDimLst	HyTime Dimension List Notation
HyDimSpc	HyTime Dimension Specification Notation
HyFunk	HyTime Marker Function Language
markfun	Marker Function Facility
valueref	Value Reference Facility

[Location Address Module]

agrovdef	Auxiliary Grove Definition Facility
bibloc	Bibliographic Location Address Facility
dataloc	Data Location Address Facility
datatok	Data Tokenizer Grove Construction Process
grovedef	Grove Definition Facilities
grovplan	Grove Plan Specification Facility
HyLexTok	HyTime Lexical Tokenizer
listloc	List Location Address Facility
mixedloc	Mixed Location Address Facility
multloc	Multiple Location Address Facility
nameloc	Name Location Address Facility
nmsploc	Name-space Location Address Facility
pathloc	Path Location Address Facility
pgrovdef	Primary Grove Definition Facility
proploc	Property Location Address Facility
queryloc	Query Location Address Facility
refctl	Reference Control Facility
referatt	Location Source Attribute of Referrer Facility
refloc	Reference Location Address Facility
reftype	Reference Typing Facility
relloc	Relative Location Address Facility
spanloc	Span Location Address Facility
treecom	Tree Combination Facility
treeloc	Tree Location Address Facility
treetype	Tree Type Specification Facility

[Hyperlink Module]

agglink	Aggregation Hyperlink Facility
anchloc	Anchor Location Address Facility

```
clink     Contextual Hyperlink Facility
hylink    Hyperlink Relationship Facility
ilink     Independent Hyperlink Facility
linkloc   Hyperlink Location Address Facility
traverse  Hyperlink Traversal Rule Specification Facility
varlink   Variable Hyperlink Facility

[Scheduling Module]

calibrat  Axis Calibration Facility
calspec   Calendar Specification Marker Function
dimref    Dimension Referencing Facility
fcsloc    Finite Coordinate Space Location Address Facility
grpdex    Group Derived Extent Specification Facility
grprepet  Group Repetition Facility
HyCalSpc  HyTime Calendar Specification Notation
HyExSpec  HyTime Extent Specification Notation
HyExtLst  HyTime Extent List Notation
HyGrand   HyTime Granule Definition Notation
measure   Measurement Domain Definition Facility
objalign  Object Alignment Facility
pulsemap  Pulse Map Specification Facility
sched     Scheduling Facility

[Rendition Module]

baton     Baton Projection Facility
batonseq  Baton Sequence Projection Facility
HyDimPro  HyTime Dimension Projector Notation
HyExPro   HyTime Extent Projector Notation
HyPro     HyTime Projector Notation
modify    Object Modification Facility
patch     Modifier Patch Facility
project   Event Projection Facility
proseq    Projector Sequence Facility
wand      Wand Modification Facility
wndpatch  Wand Patch Facility

-->

<?IS10744 USELEX HyTimelx>
<!entity %
   HyTimelx        -- HyTime lexical types --

   PUBLIC "ISO/IEC 10744:1997//NONSGML LTDR LEXTYPES
           HyTime Lexical Types//EN"
>

                     <!-- Combination options -->

<!entity % grovedef "IGNORE">
<![ %grovedef; [
   <!entity % agrovdef "INCLUDE">
   <!entity % pgrovdef "INCLUDE">
]]>
```

```
          <!-- General Architecture Support Declarations -->

<!-- The following declarations ensure propagation of user-specified
     General Architecture options. -->

<!entity % altreps "IGNORE">
<![ %altreps; [
   <!entity % gaaltrp "altreps">
]]>
<!entity % gaaltrp "">

<!entity % dafe "IGNORE">
<![ %dafe; [
   <!entity % gadafe "dafe">
]]>
<!entity % gadafe "">

<!entity % dvlist "IGNORE">
<![ %dvlist; [
   <!entity % gadvl "dvlist">
]]>
<!entity % gadvl "">

<!entity % HyLex "IGNORE">
<![ %dafe; [
   <!entity % gaHyLex "HyLex">
]]>
<!entity % gaHyLex "">

<!entity % HyOrd "IGNORE">
<![ %dafe; [
   <!entity % gaHyOrd "HyOrd">
]]>
<!entity % gaHyOrd "">

<!entity % included "IGNORE">
<![ %included; [
   <!entity % gaincl "included">
]]>
<!entity % gaincl "">

<!entity % ireftype "IGNORE">
<![ %ireftype; [
   <!entity % gairftp "ireftype">
]]>
<!entity % gairftp "">

<!entity % lextype "IGNORE">
<![ %lextype; [
   <!entity % galextp "lextype">
]]>
<!entity % galextp "">

<!entity % opacity "IGNORE">
<![ %opacity; [
   <!entity % gaopaci "opacity">
]]>
<!entity % gaopaci "">

<!entity % REGEX "IGNORE">
<![ %dafe; [
   <!entity % gaREGEX "REGEX">
]]>
<!entity % gaREGEX "">

<!entity % superdcn "IGNORE">
<![ %superdcn; [
   <!entity % gaspdcn "superdcn">
]]>
<!entity % gaspdcn "">

<?IS10744 ArcBase GenArc>

<!NOTATION GenArc
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           General Architecture//EN"
>
<!ATTLIST #NOTATION GenArc
   ArcFormA NAME     #FIXED GAForm
   ArcNamrA NAME     #FIXED GANames
   ArcSuprA NAME     #FIXED GASupr
   ArcIgnDA NAME     #FIXED GAIgnD
   ArcDocF  NAME     #FIXED GADoc
   ArcDTD   CDATA    #FIXED "%GenArc"
   ArcDataF NAME     #FIXED GABridN
   ArcBridF NAME     #FIXED GABrid
   ArcAuto  NMTOKEN  #FIXED ArcAuto
   ArcOptSA NAMES    #FIXED "commatts dcnatts"
   commatts CDATA    #FIXED "%gadafe; %gadvl; %gaHyLex; %gaHyOrd;
                             %gairftp; %galextp; %gaopaci; %gaREGEX;"
   dcnatts  CDATA    #FIXED "%gaaltrp; %gaincl; %gaspdcn;"
>
<!entity % GenArc
   PUBLIC "ISO/IEC 10744:1997//DTD AFDR Meta-DTD
           General Architecture//EN"
>

<!-- This includes the General Architecture meta-DTD, making all of
     its facilities available through HyTime. -->

%GenArc;


          <!-- HyTime Architecture Support Declarations -->

<!-- These support declarations allow some HyTime forms to be defined
     in terms of other HyTime forms, saying that, for instance, a
     "clink" is a kind of "hylink", or that a "nameloc" is a kind of
     "mixedloc". -->

<!-- The following declarations include the HyBase facilities needed
     to support specified user options. -->
```

```
<!entity % conloc "IGNORE">
<![ %conloc; [
   <!entity % hbvalref "valueref">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % dimspec "IGNORE">
<![ %dimspec; [
   <!entity % hbmrkfun "markfun">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % multloc "IGNORE">
<!entity % nameloc "IGNORE">
<!entity % HyLexTok "IGNORE">
<!entity % dataloc "IGNORE">
<!entity % datatok "IGNORE">
<!entity % queryloc "IGNORE">
<!entity % referatt "IGNORE">
<!entity % spanloc "IGNORE">
<!entity % treetype "IGNORE">
<![ %dataloc; [
   <![ %referatt; [
      <!entity % hbrefrat "referatt">
   ]]>
   <![ %multloc; [
      <!entity % hbmultlc "multloc">
   ]]>
   <![ %spanloc; [
      <!entity % hbspanlc "spanloc">
      <![ %treetype; [
         <!entity % hbtreetp "treetype">
      ]]>
   ]]>
   <![ %HyLexTok; [
      <!entity % hbhylxtk "HyLexTok">
   ]]>
   <!entity % hbdattok "datatok">
   <!entity % HyBase "INCLUDE">
]]>

<![ %HyLexTok; [
   <!entity % hbdattok "datatok">
   <!entity % HyBase "INCLUDE">
]]>

<![ %datatok; [
   <!entity % hbagrvdf "agrovdef">
   <!entity % HyBase "INCLUDE">
]]>

<![ %nameloc; [
   <![ %queryloc; [
      <!entity % hbqryloc "queryloc">
   ]]>
   <![ %referatt; [
      <!entity % hbrefrat "referatt">
   ]]>
   <![ %multloc; [
      <!entity % hbmultlc "multloc">
   ]]>
   <![ %spanloc; [
      <!entity % hbspanlc "spanloc">
      <![ %treetype; [
         <!entity % hbtreetp "treetype">
      ]]>
   ]]>
   <!entity % hbmixloc "mixedloc">
   <!entity % hbnmsplc "nmsploc">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % linkloc "IGNORE">
<![ %linkloc; [
   <![ %referatt; [
      <!entity % hbrefrat "referatt">
   ]]>
   <![ %multloc; [
      <!entity % hbmultlc "multloc">
   ]]>
   <![ %spanloc; [
      <!entity % hbspanlc "spanloc">
      <![ %treetype; [
         <!entity % hbtreetp "treetype">
      ]]>
   ]]>
   <!entity % hbqryloc "queryloc">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % anchloc "IGNORE">
<![ %anchloc; [
   <![ %referatt; [
      <!entity % hbrefrat "referatt">
   ]]>
   <![ %multloc; [
      <!entity % hbmultlc "multloc">
   ]]>
   <![ %spanloc; [
      <!entity % hbspanlc "spanloc">
      <![ %treetype; [
         <!entity % hbtreetp "treetype">
      ]]>
   ]]>
   <!entity % hbqryloc "queryloc">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % clink "IGNORE">
<!entity % traverse "IGNORE">
<![ %clink; [
   <![ %traverse; [
      <!entity % hbtrav "traverse">
   ]]>
   <!entity % hbhylink "hylink">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % agglink "IGNORE">
<![ %agglink; [
   <![ %traverse; [
      <!entity % hbtrav "traverse">
   ]]>
   <!entity % hbhylink "hylink">
   <!entity % hbrefloc "refloc">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % dimref "IGNORE">
<![ %dimref; [
   <!entity % hbmrkfun "markfun">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % calspec "IGNORE">
<![ %calspec; [
   <!entity % hbmrkfun "markfun">
   <!entity % HyBase "INCLUDE">
]]>

<!entity % hbmrkfun "">
<!entity % hbvalref "">
<!entity %
   hbbase         -- HyBase base module options --

   '"%hbmrkfun; %hbvalref;"'
>

<!entity % hbagrvdf "">
<!entity % hbmixloc "">
<!entity % hbnmsplc "">
<!entity % hbhylxtk "">
<!entity % hbdattok "">
<!entity % hbqryloc "">
<!entity % hbrefloc "">
<!entity % hbrefrat "">
<!entity % hbmultlc "">
<!entity % hbspanlc "">
<!entity % hbtreetp "">
<!entity %
   hblocs         -- HyBase location address module options --

   '"%hbagrvdf; %hbmixloc; %hbnmsplc; %hbhylxtk; %hbdattok; %hbqryloc;
     %hbrefloc; %hbrefrat; %hbmultlc; %hbspanlc; %hbtreetp;"'
>

<!entity % hbhylink "">
```

```
<!entity % hbtrav "">
<!entity %
   hblinks        -- HyBase hyperlinking module options --

   '"%hbhylink; %hbtrav;"'
>

<!entity % HyBase "IGNORE">
<![ %HyBase; [
<?IS10744 ArcBase HyBase>

<!NOTATION HyBase
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<!ATTLIST #NOTATION HyBase
   ArcFormA NAME     #FIXED HyBase
   ArcNamrA NAME     #FIXED HyBNames
   ArcDocF  NAME     #FIXED HyDoc
   ArcDTD   CDATA    #FIXED "HyBase"
   ArcDataF NAME     #FIXED HyBridN
   ArcBridF NAME     #FIXED HyBrid
   ArcAuto  NMTOKEN  #FIXED nArcAuto
   ArcOptSA NAMES    #FIXED "GenArc base locs links sched rend"
   GenArc   CDATA    #IMPLIED
   base     CDATA    #FIXED %hbbase;
   locs     CDATA    #FIXED %hblocs;
   links    CDATA    #FIXED %hblinks;
   sched    CDATA    #IMPLIED
   rend     CDATA    #IMPLIED
>

<!NOTATION AFDRMeta
   PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD Notation//EN"
>
<!ENTITY HyBase
   PUBLIC "ISO/IEC 10744:1997//DTD AFDR Meta-DTD
           Hypermedia/Time-based Structuring Language (HyTime)//EN"
   CDATA AFDRMeta
>
]]><!-- HyBase -->

              <!-- Content model parameter entities -->
<!entity %
   HyCFC          -- HyTime context-free content --
                  -- Note: %loc, %link, %resorce qualify but are used
                     as meta-inclusions rather than meta-proper-
                     subelements --

   "%GACFC;|HyBrid|fcs"
>
<!entity %
   loc            -- Location address forms --

   "anchloc|bibloc|dataloc|fcsloc|linkloc|listloc|mixedloc|
    nameloc|nmsploc|pathloc|proploc|queryloc|relloc|treeloc"
```

```
>
<!entity %
   link           -- Hyperlink forms --

   "agglink|clink|hylink|ilink|varlink"
>
<!entity %
   resbase        -- Base module resource forms --

   "actrule|bosspec|desctab"
>
<!entity %
   resloc         -- Location address module resource forms --

   "agrovdef|datatok|grovplan|pgrovdef"
>
<!entity %
   resschd        -- Scheduling module resource forms --

   "calspec|evsched|extent|extlist|measure"
>
<!entity %
   resrend        -- Rendition module resource forms --

   "baton|batrule|batseq|modpatch|modrule|projectr|prorule|proseq|
    rendrule|wandrule|wand|wndpatch"
>
<!entity %
   resorce        -- All resource architectural forms --

   "%resbase;|%resloc;|%resschd;|%resrend;"
>
<!entity %
   marklist       -- Axis marker list content model --
                  -- Clause: 6.8.1.1 --
                  -- Data and elements that resolve to lists of
                     markers. --

   "#PCDATA|dimref|markfun"
>
<!entity %
   dimlist        -- HyTime Dimension List content --
                  -- Clause: 6.8.4 --
   "%marklist;|dimspec"
>

<!element
   HyDoc          -- HyTime document element --
                  -- Clause: 6.4 --
   - O
   (%HyCFC;)*
   +(%link;|%loc;|%resorce;)

-- OptionalAttributes [base]: bos, bosspcat --
-- OptionalAttributes [locs]: dgrvplan --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>

            <!-- HyTime Architectural Bridging Element -->
<!element
   HyBrid         -- HyTime architectural bridging element --
                  -- Clause: 6.6 --

   - O
   (%HyCFC;)*

-- Attributes [base]: HyBrid --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   HyBrid         -- HyTime architectural bridging element --
                  -- Clause: 6.6 --

   GenArc   NAME     #FIXED GABrid
>

           <!-- HyTime Architectural Bridging Notation -->
<!notation
   HyBridN        -- HyTime architectural bridging notation --
                  -- Clause: 6.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Architectural Bridging Notation//EN"

-- Attributes [base]: HyBridN --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyBridN        -- HyTime architectural bridging notation --
                  -- Clause: 6.6 --

   GenArc   NAME     #FIXED GABridN
>


<!entity % bosspec "IGNORE">

<![ %bosspec; [
<!attlist
-- bosspcat --    -- BOS except specification attributes --
                  -- Clause: 6.5.3 --
   (HyDoc)
   bosspec        -- Bounded object set exception specification --
                  -- Adjustments to be made to the bounded object
```

```
                     set. --
      IDREFS      -- Reference --
                  -- Reftype: bosspec+ --
                  -- Constraint: must be internal reference --
      #IMPLIED    -- Default: no BOS exception specification --
>
]]><!-- bosspec -->

         <!-- Bounded Object Set Exception Specification -->
<![ %bosspec; [
<!element
   bosspec        -- Bounded object set exception specification --
                  -- Clause: 6.5 --
                  -- Used to affect the HyTime BOS by overriding the
                     inclusion or exclusion and priority of the
                     entities identified by the BOS path or paths
                     given as content. --
   - O
   (#PCDATA)      -- Lextype: ((ENTITY,(csname|literal)*)|
                               (GRPO,ENTITY,(csname|literal)*,GRPC)+)
                     --
                  -- Constraint: If parentheses are used, each
                     parenthesized list is a separate BOS path. --
                  -- Constraint: Each word or literal in a BOS path is
                     the name of an entity declared in the entity
                     identified by the previous word, literal, or
                     entity name. --

-- Attributes [base]: bosspec --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: HyDoc:bosspec --
>
<!attlist
   bosspec        -- Bounded object set exception specification --
                  -- Clause: 6.5 --

   boslevel       -- BOS level --
                  -- The BOS level from the last entity named in
                     each specified BOS path to be affected by this
                     bosspec. --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=last entity only) --
      1

   inbos          -- Include in BOS --
                  -- Unconditionally include or exclude objects
                     declared by the last entity named in each BOS
                     path, to the BOS level specified by this
                     bosspec's boslevel attribute. --
      (inbos|notinbos)
      #IMPLIED    -- Default: BOS unaffected --

   bosprrty       -- Bounded object set priority --
```

```
                  -- Unconditionally specify the BOS priority of
                     objects declared by the last entity named in each
                     BOS path, to the BOS level specified by this
                     bosspec's boslevel attribute. --
                  -- Note: The semantic of the bosprrty attribute is
                     not affected by the value of the inbos attribute
                     (that is, whether it is explicitly "inbos" or the
                     value is implied). --
      (foregrnd|backgrnd)
      #IMPLIED    -- Default: no priority change --
>
]]><!-- bosspec -->


<!entity % bos "IGNORE">

<![ %bos; [
<!attlist
   -- bos --      -- HyTime bounded object set --
                  -- Clause: 6.5.1 --
  (HyDoc)

   maxbos         -- Maximum bounded object set level --
                  -- Bounding level of HyTime bounded object set when
                     document is a hub or subhub. --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=hub only) --
      0

   boslevel       -- Bounded object set level --
                  -- Default BOS level used by data entities declared
                     in hub document. --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=this entity only) --
      #IMPLIED    -- Default: No HyTime BOS --

>
]]><!-- bos -->

             <!-- HyTime BOS Control Data Attributes -->
<![ %bos; [
<!attlist #NOTATION
-- bosdatt --     -- HyTime BOS control data attributes --
                  -- Clause: 6.5.2 --
   #ALL

   boslevel       -- BOS level --
                  -- Bounded object set level for the entity --
      NUMBER      -- Constraint: depth of nested entities to include
                     in BOS (0=no limit, 1=this entity only) --
      #IMPLIED    -- Default: value of boslevel attribute of HyDoc
                     element. --

   inbos          -- Include in BOS --
                  -- Unconditional include in, or exclude from, BOS --
      (inbos|notinbos)
      #IMPLIED    -- Default: inclusion controlled by BOS level --

   bosprrty       -- Bounded object set priority --
                  -- Default BOS priority of objects in entity tree
                     rooted at this entity. --
      (foregrnd|backgrnd)
      foregrnd

   subhub         -- Is entity a subhub? --
      (subhub|nosubhub)
      nosubhub
>
]]><!-- bos -->


<!entity % valueref "IGNORE">

             <!-- Value Reference Attribute Namer -->
<![ %valueref; [
<!attlist
-- valueref --    -- Value reference attribute namer --
                  -- Clause: 6.7.1 --
   #ALL

   valueref       -- Value reference attribute namer --
                  -- Associates attributes (or element's content) with
                     referential attributes (or content) that can be
                     used to refer to the semantic value of the
                     attribute or content. --
      CDATA       -- Lextype: ((ATTORCON|"#ELEMENT"),ATTORCON)+ --
                  -- Note: The first ATTORCON is the attribute or
                     content for which the value is to be addressed,
                     the second ATTORCON is the attribute or content
                     by which the value reference is made. --
                  -- Constraint: second attribute in each pair must be
                     a referential attribute or content defined as
                     referential by the refloc facility. --
                  -- Constraint: "#ELEMENT" and "#CONTENT" may only
                     occur once as first keyword of pair.  "#CONTENT"
                     may only occur once as second keyword of pair. --
      #IMPLIED    -- Constant --
>
]]><!-- valueref -->


<!entity % HyExSpec "IGNORE">

            <!-- HyTime Extent Specification Notation -->
<![ %HyExSpec; [
<!notation
   HyExSpec       -- HyTime Extent Specification Notation --
                  -- Clause: 9.4.4 --
                  -- A list of dimensions (in the HyTime Dimension
                     List Notation, after resolution of marker
                     functions) that together specify a single extent
                     on one or more axes. --
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Dimension Specification Notation//EN"

-- Attributes [sched]: HyExSpec --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyExSpec       -- HyTime Extent Specification Notation --
                  -- Clause: 9.4.4 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyDimLst
>
<!entity % HyDimLst "INCLUDE" >
<!entity % sched "INCLUDE">
]]><!-- HyExSpec -->


<!entity % HyExtLst "IGNORE">

                 <!-- HyTime Extent List Notation -->
<![ %HyExtLst; [
<!notation
   HyExtLst       -- HyTime Extent List Notation --
                  -- A list of extents each of which may be specified
                     as an extent, part of another extent list, or or
                     as all or part of a list of axis markers
                     explicitly entered or returned by a marker
                     function. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Extent List Notation//EN"

-- Attributes [sched]: HyExtLst --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyExtLst       -- HyTime Extent List Notation --
                  -- Clause: 9.4.6 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyDimLst
>
<!entity % HyDimLst "INCLUDE" >
<!entity % sched "INCLUDE">
]]><!-- HyExtLst -->


<!entity % HyDimSpc "IGNORE">

                   <!-- Dimension specification -->
```

```
<![ %dimspec; [
<![ %HyDimSpc; [
   <!entity % ddimspec "HyDimSpc">
]]>
<!entity %
   ddimspec       -- Default dimension specification notation --
                  -- Clause: 6.8.3 --

   "#REQUIRED"
>
<!element
   dimspec        -- Dimension specification --
                  -- Clause: 6.8.3 --
                  -- A position and quantum count along a single
                     axis. When used as a marker function, returns a
                     pair of positive axis markers, the first being
                     the position of the first quantum of the
                     dimension and the second being the quantum count
                     of the dimension. --
   O O
   (%HyCFC;|%marklist;)*
                  -- Constraint: Content must conform to the notation
                     specified in the dimspec's notation attribute. --

-- Attributes [base]: dimspec --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dimspec        -- Dimension specification --
                  -- Clause: 6.8.3 --

   HyBase   NAME     #FIXED markfun

   notation       -- Dimension specification notation --
      NAME        -- Lextype: NOTATION --
      %ddimspec;
>
]]><!-- dimspec -->


<!entity % fcsloc "IGNORE">

          <!-- Finite coordinate space location address -->
<![ %fcsloc; [
<![ %HyExtLst; [
   <!entity % dfcslcnt "HyExtLst">
]]>
<!entity %
   dfcslcnt       -- Default fcsloc extent list notation --
                  -- Clause: 9.10 --

   "#REQUIRED"
>
```

```
<!element
   fcsloc         -- Finite coordinate space location address --
                  -- Clause: 7.10.2 --
                  -- Addresses events, modscopes, and proscopes (or
                     the objects, modifiers, and projectors they
                     contain) in finite coordinate spaces. --
                  -- Constraint: location source must be an FCS, event
                     schedule, wand, or baton --
   - O
   (%dimlist;|extent|extlist)*
                  -- Constraint: if no extent list notation is
                     specified, content must consist of only extent
                     and extlist elements. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- Attributes [sched]: fcsloc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- OptionalAttributes [sched]: rfcsloc --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   fcsloc         -- Finite coordinate space location address --
                  -- Clause: 9.10 --

   HyBase   NAME     #FIXED queryloc

   notation       -- Extent list notation for selection --
      NAME        -- Lextype: NOTATION --
      %dfcslcnt;  -- Default: content must consist of only extent and
                     extlist elements. --

   select         -- Precision of selection --
                  -- Description: How much of an event, modscope, or
                     proscope must appear within the selection
                     boundary in order for it (or the object,
                     modifier, or proscope that it schedules) to be
                     selected. --
      CDATA       -- Lextype: ("#ALL"|"#ANY"|(unzi,granule?))+ --
                  -- Constraint: one for each axis of location source
                     or one for all axes. --
                  -- Constraint: Precision unzi is specified in terms
                     of the following granule, if supplied; otherwise
                     in terms of the axis HMU defined for the schedule. --
      "#ALL"      -- Default: event, modscope, or proscope must be
                     entirely within selection boundary in order to be
                     selected. --

   objects        -- Events or objects? --
                  -- Node type to select: events, modscopes, or
                     proscopes (events) or the objects, modifiers, and
                     projectors that are scheduled by them
                     (objects)? --
      (events|objects)
      events

   dsdtypes       -- Directly scheduled types to select --
                  -- Types of directly scheduled elements to be
                     selected. --
      CDATA       -- Lextype: ("#ALL"|("#EVENT"|"#MODSCOP"|
                                      "#PROSCOP"|GI)*) --
                  -- Constraint: Can be any combination of #EVENT,
                     #MODSCOP, #PROSCOP, and the GIs of event,
                     modscope or proscope form elements.  Can also be
                     #ALL. --
      #IMPLIED    -- Default: For each form of element used as locsrc:
                       locsrc element:       default value:
                       - - - - - - -         - - - - - - -
                         evsched               "#EVENT"
                         wand                  "#MODSCOP"
                         baton                 "#PROSCOP"
                         fcs                   "#ALL"
                     --
>
<!entity % sched "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- fcsloc -->

                   <!-- Calendar Specification -->
<![ %calspec; [
<!notation
   HyCalFun       -- HyTime Calendar Marker Function Notation --
                  -- Clause: 9.9.2 --
                  -- A calendar specification is a marker function to
                     be used to specify a quantum along a coordinate
                     axis whose measurement domain is real time (that
                     is, SIsecond) and that has been calibrated using
                     HyCalSpc notation.  If used as the first marker
                     of a dimension specification, a positive marker
                     is returned.  If used as the second marker of a
                     dimension specification, a negative marker is
                     returned. --
                  -- Constraint: highest-order date/time component
                     specified cannot exceed that specified for axis
                     calibration. --
                  -- Constraint: omitted date/time components are
                     those of previous specified calspec for this
                     schedule. --
                  -- Lextype: ((JULDATE|fulldate|monthday|day)?,
                               (UTCtime|hrminsec|hrmin|minute)?,
                               timeoff*) --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Calendar Marker Function Notation//EN"

-- Attributes [sched]: HyCalFun --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyCalFun       -- HyTime Calendar Marker Function Notation --
                  -- Clause: 9.9.2 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyCalSpc

   GMTorUTC       -- GMT or UTC timescale --
                  -- Note: (GMT noon = UTC midnight) --
      (GMT|UTC)
      UTC
>
<!element
   calspec        -- Calendar Specification --
                  -- Clause: 9.9.2 --
   O O
   (%HyCFC;)*     -- Lextype: ((JULDATE|fulldate|monthday|day)?,
                             (UTCtime|hrminsec|hrmin|minute)?,
                             timeoff*) --
                  -- Constraint: If neither JULDATE nor fulldate is
                     present, previously specified year and other
                     larger quanta will be assumed. --

-- Attributes [sched]: calspec --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   calspec        -- Calendar Specification --
                  -- Clause: 9.9.2 --

   HyBase   NAME     #FIXED markfun
   notation NOTATION (HyCalFun) #FIXED HyCalFun

   GMTorUTC       -- GMT or UTC timescale --
                  -- Note: (GMT noon = UTC midnight) --
      (GMT|UTC)
      UTC
>
<!entity % HyCalSpc "INCLUDE">
]]><!-- calspec -->

                <!-- Explicit Dimension Reference -->
<![ %dimref; [
<!notation
   dimref         -- Explicit dimension reference marker function
                     notation --
                  -- Clause: 9.8.2.1 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Explicit dimension reference marker function notation//EN"
```

```
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!element
   dimref         -- Explicit dimension reference --
                  -- Clause: 9.8.2.1 --
   - O
   (%HyCFC;)*

-- Attributes [sched]: dimref --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dimref         -- Explicit dimension reference --
                  -- Clause: 9.8.2.1 --

   HyBase   NAME     #FIXED markfun
   notation NOTATION (dimref) #FIXED dimref

   elemspec       -- Element specified --
                  -- Element from which referenced dimension is
                     derived --
      CDATA       -- Reference --
                  -- Reftype: (baton|event|evgrp|evsched|extent|
                              extlist|fcs|modgrp|modscope|progrp|
                              proscope|wand) --
      #REQUIRED

   selcomp        -- Selected component of dimension --
                  -- Dimension component of specified element to be
                     referenced.
                        first:    First or lowest quantum
                        last:     Last or highest quantum
                        qcnt:     Quantum count --
      (first|last|qcnt)
      qcnt

   flip           -- Flip selected component --
                  -- Whether or not to invert the referenced dimension
                     component.  Specifically:
                        noflip:   First is reported as a positive
                                  integer corresponding to the number
                                  of the first quantum counting from
                                  the beginning of the axis, last is
                                  reported as a negative integer
                                  corresponding to the number of the
                                  last quantum counting from the end of
                                  the axis, and qcnt is reported as a
                                  positive integer.
                        flip:     First is reported as a negative
                                  integer corresponding to the number
```

```
                             of first quantum counting from the
                             end of the axis, last is reported as
                             a positive integer corresponding to
                             the number of the last quantum
                             counting from the beginning of the
                             axis, and qcnt is reported as a
                             negative integer. --
   (flip|noflip)
   noflip

schdspec       -- Element with sched attribute form specified --
               -- Establishes context in which extents are
                  interpreted: FCS, axis order, and HMU --
               -- Constraint: Ignored if element specified is
                  evsched, wand, baton or fcs. --
   CDATA       -- Reference --
               -- Reftype: (baton|evsched|wand)? --
   #IMPLIED    -- Default: dimref is treated as if the containing
                  element with the sched attribute form is
                  referenced by schdspec. --
               -- Constraint: Required if elemspec is extent or
                  extlist. --

axisspec       -- Axis specified --
               -- Axis on which referenced dimension occurs --
   NAME        -- Lextype: AXISNM --
               -- Constraint: must be valid axis name of FCS in
                  which the referenced dimension occurs. --
   #IMPLIED    -- Default: only axis --

dimnum         -- Dimension number --
   NUMBER      -- Constraint: 1 = first dimension in order of
                  appearance on axis --
   #IMPLIED    -- Default: smallest dimension encompassing all
                  dimensions --

extnum         -- Extent number --
   NUMBER      -- Constraint: 1 = first or only extent in order of
                  specification/appearance in document --
   #IMPLIED    -- Default: combination of all relevant extents --

granule        -- Granule for reporting --
               -- Granule for reporting referenced dimension --
   CDATA       -- Lextype: granule --
               -- Constraint: granule must be defined in terms of
                  the same SMU as axis --
   #IMPLIED    -- Default: MDU of axis of referenced dimension --

roffgran       -- Round off to granule handling --
               -- Specifies how to handle roundoff.
                     roerr:    Report error if rounding is required.
                     rocmplet: Round off to minimum complete
                               dimension; that is, minimum dimension
                               expressible in terms of the granule
                               specified that completely covers the
                               actual dimension.
```

```
                        roscant:  Round off to maximum complete
                                  dimension; that is, maximum dimension
                                  expressible in terms of the granule
                                  specified that is fully occupied by
                                  the actual dimension
                        roboundy: Round off to dimension boundaries;
                                  that is, dimension that places the
                                  boundaries of the reported dimension
                                  as close as possible to the
                                  boundaries of the actual
                                  dimension. --
      (roboundy|rocmplet|roerr|roscant)
      roboundy

   dsdtypes       -- Directly scheduled types for calculating
                     referenced dimension --
                  -- Types of directly scheduled elements to be taken
                     into account when calculating the referenced
                     dimension. --
      CDATA       -- Lextype: ("#ALL"|("#BATON"|"#EVENT"|"#EVGRP"|
                                  "#EVSCHED"|"#EXTENT"|
                                  "#EXTLIST"|"#FCS"|"#MODGRP"|
                                  "#MODSCOP"|"#PROGRP"|
                                  "#PROSCOP"|"#WAND"|GI)*) --
                  -- Constraint: GIs must be of event, modscope,
                     proscope, evgrp, modgrp, progrp, evsched, wand,
                     baton, or fcs form elements. --
      #IMPLIED    -- Default: Depends on form of element specified, as
                     follows:
                      element specified:    default value:
                      - - - - - - - - -     - - - - - - -
                      extent                (null string)
                      extlist, event,
                      modscope or proscope  "#EXTENT #EXTLIST"
                      evgrp or evsched      "#EXTENT #EXTLIST #EVENT
                                             #EVGRP"
                      modgrp or wand        "#EXTENT #EXTLIST #MODSCOP
                                             #MODGRP"
                      progrp or baton       "#EXTENT #EXTLIST #PROSCOP
                                             #PROGRP"
                      fcs                   "#ALL" --
>
<!entity % sched "INCLUDE">
]]><!-- dimref -->

              <!-- Hyperlink Anchor Location Address -->
<![ %anchloc; [
<!notation
   HyAncLoc       -- HyTime hyperlink anchor queryloc notation --
                  -- Clause: 8.3.2 --
                  -- Constraint: location source objects must be
                     hyperlinks or documents --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" (and referrer is not a
                     hyperlink) location source is document in which
                     the primary tree root or referrer occur. --
                  -- Lextype: (csname|literal)* --
                  -- Constraint: csnames and literals are anchor
                     role names of links in location source. --
                  -- Constraint: if no anchor roles specified,
                     addresses all anchors of links in location
                     source. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Hyperlink Anchor Queryloc Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!element
   anchloc        -- Hyperlink anchor location address --
                  -- Clause: 8.3.2 --
                  -- Constraint: location source objects must be
                     hyperlinks or documents --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" (and referrer is not a
                     hyperlink) location source is document in which
                     the primary tree root or referrer occur. --
   - O
   (#PCDATA)      -- Lextype: (csname|literal)* --
                  -- Constraint: csnames and literals are anchor
                     role names of links in location source. --
                  -- Constraint: if no anchor roles specified,
                     addresses all anchors of links in location
                     source. --

-- Attributes [locs]: anchloc, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   anchloc        -- Hyperlink anchor location address --
                  -- Clause: 8.3.2 --

   HyBase   NAME     #FIXED queryloc
   notation NOTATION (HyAncLoc) #FIXED HyAncLoc

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- anchloc -->

                 <!-- Hyperlink location address -->
<![ %linkloc; [
<!notation
```

```
   HyLnkLoc       -- HyTime hyperlink queryloc notation --
                  -- Clause: 8.3.1 --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" location source is
                     document in which the primary tree root or
                     referrer occur. --
                  -- Lextype: (csname|literal)+ --
                  -- Constraint: csnames and literals must be link
                     types (GIs) in location source. --
                  -- Constraint: if no link types specified, addresses
                     all links in location source. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Hyperlink Queryloc Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!element
   linkloc        -- Hyperlink location address --
                  -- Clause: 8.3.1 --
                  -- Constraint: if implied location source is
                     "ptreert" or "referrer" location source is
                     document in which the primary tree root or
                     referrer occur. --
   - O
   (#PCDATA)      -- Lextype: (csname|literal)+ --
                  -- Constraint: csnames and literals must be link
                     types (GIs) in location source. --
                  -- Constraint: if no link types specified, addresses
                     all links in location source. --

-- Attributes [locs]: linkloc, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   linkloc        -- Hyperlink location address --
                  -- Clause: 8.3.1 --

   HyBase   NAME     #FIXED queryloc
   notation NOTATION (HyLnkLoc) #FIXED HyLnkLoc

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- linkloc -->

                   <!-- Name List Specification -->
```

```
<![ %nameloc; [
<!element
   nmlist         -- Name list specification --
                  -- Clause: 7.9.6 --
                  -- Addresses elements or entities in an SGML document --
   - O
   (#PCDATA)      -- Reference --
                  -- Lextype: (IDREF|ENTITY)* --

-- Attributes [locs]: nmlist, proplat, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nmlist         -- Name list specification --
                  -- Clause: 7.9.6 --

   HyBase   NAME     #FIXED nmsploc
   impsrc   NAME     #FIXED grovert
   HyBnames CDATA    #FIXED "locsrc docorsub
                             namespc nametype
                                #MAPTOKEN elements element
                                #MAPTOKEN entities entity"

   nametype       -- Name-space from which nodes are selected --
      (entity|element)
      entity

   docorsub       -- Document or subdocument location source --
      ENTITY
      #IMPLIED

   notspace       -- If nmspace name is invalid? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   notname        -- If name is not valid in nmspace? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % proplat "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- nameloc -->

              <!-- Name List Query Location Address -->
<![ %nameloc; %queryloc; [
<!element
   nmquery        -- Name list query location address --
                  -- Clause: 7.11.2 --
                  -- Locates elements or entities by querying their
                     properties. --
   - O
   (%HyCFC;)*

-- Attributes [locs]: locsrc, nmquery, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nmquery        -- Name list query location address --
                  -- Clause: 7.11.2 --

   HyBase   NAME     #FIXED queryloc
   HyBnames CDATA    #FIXED "locsrc qdomain"

   qdomain        -- Query domain --
      CDATA       -- Reference --
      #IMPLIED    -- Default: implied as described for impsrc --

   notation       -- Query notation --
      NAME        -- Lextype: NOTATION --
      #REQUIRED

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- nmquery -->

                   <!-- Named Location Address -->
<![ %nameloc; [
<!element
   nameloc        -- Named Location Address --
                  -- Clause: 7.9.5 --
                  -- Assigns a local ID to one or more named objects --
   - O
   (nmlist|nmquery)*

-- Attributes [locs]: nameloc --
-- OptionalAttributes [locs]: multloc, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nameloc        -- Named Location Address --
                  -- Clause: 7.9.5 --

   HyBase   NAME     #FIXED mixedloc
>
]]><!-- nameloc -->
```

```
                       <!-- Contextual Link -->
<![ %clink; [
<!element
   clink          -- Contextual link --
                  -- Clause: 8.2.2 --
   - O
   (%HyCFC;)*

-- Attributes [links]: clink --
-- OptionalAttributes [links]: clinktra --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   clink          -- Contextual link --
                  -- Clause: 8.2.2 --

   HyBase   NAME     #FIXED hylink
   HyBnames CDATA    #FIXED "refsub linkend"
   anchrole CDATA    #FIXED "refmark refsub #LIST"
   anchcstr NAMES    "self required"

   linkend        -- Reference subject link end --
      CDATA       -- Reference --
      #IMPLIED    -- Default: omitted or conditional --
                  -- Constraint: if left unspecified, must specify
                     COND in anchcstr --
>
]]><!-- clink -->

                      <!-- Aggregation Link -->
<![ %agglink; [
<!element
   agglink        -- Aggregation link --
                  -- Clause 8.2.3 --
   - O
   (%HyCFC;)*

-- Attributes [links]: agglink
-- OptionalAttributes [links]: traverse --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   agglink        -- Aggregation link --
                  -- Clause 8.2.3 --

   HyBase   NAME     #FIXED hylink
   anchrole CDATA    #FIXED "agg members #LIST"
   anchcstr NAMES    "self required"
   loctype  CDATA    "members IDLOC"
   members        -- Members of aggregate --
      CDATA       -- Reference --
                  -- Note: Because members is by default given the
                     location address type IDLOC, its value is
                     interpreted by default as if it were IDREFS,
                     though the interpretation can be changed by
                     specifying a different value for the loctype
                     attribute. --
      #IMPLIED    -- Default: omitted or conditional --
                  -- Constraint: if left unspecified, must specify
                     COND in anchcstr --
>
]]><!-- agglink -->

                    <!-- Data Location Address -->
<![ %dataloc; [
<!element
   dataloc        -- Data location address --
                  -- Clause: 7.10.2 --
                  -- Locates string and token data objects in data --
   - O
   (%dimlist;)*   -- Constraint: interpreted as a single list of
                     dimension specifications --

-- Attributes [base]: overrun --
-- Attributes [locs]: dataloc, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dataloc        -- Data location address --
                  -- Clause: 7.10.2 --

   HyBase   NAME     #FIXED datatok
   HyBnames CDATA    #FIXED "HyBase #DEFAULT
                             #ARCCONT filter
                             grovesrc locsrc"

   filter         -- Tokenization filter --
      CDATA       -- Constraint: If no data tokenizer notation
                     specified, filter must be a single lexical type
                     name. --
      "str"

   tokordat        -- Result to return: tokens or source data? --
      (data|tokens)
      data

-- Attributes from datatok form --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- dataloc -->

                      <!-- Content Location -->
<![ %conloc; [
<!attlist
-- conloc --      -- Content Location --
                  -- Clause: 6.7.1 --
   #ALL

   HyBase   NAME     #FIXED HyBrid
   valueref CDATA    #FIXED "#CONTENT conloc"

   conloc         -- Content location --
                  -- Location of the element's semantic content. --
      CDATA       -- Reference --
      #CONREF     -- Default: syntactic content --
>
]]><!-- conloc -->


<!entity % desctxt "IGNORE">

                 <!-- Descriptive Text Attributes -->
<![ %desctxt; [
<!attlist
-- dtxtatt --     -- Descriptive text attributes --
                  -- Clause: 6.7.2.1 --
   #ALL

   desctxt        -- Descriptive text --
                  -- If specified, its descdef in desctab is treated
                     by HyTime as the element or as its content. --
      CDATA       -- Lextype: words --
      #CONREF     -- Default: syntactic content (and attributes) --

   desctab        -- Description table --
                  -- Current description tables --
      CDATA       -- Reference --
                  -- Reftype: desctab+ --
                  -- Constraint: searched in order listed --
      #IMPLIED    -- Default: use previously specified set of
                     tables --
>
]]><!-- desctxt -->

                      <!-- Description Table -->
<![ %desctxt; [
<!element
   desctab        -- Description table --
                  -- Clause: 6.7.2.2 --
                  -- Descriptive text definition table --
   - O
   (desctxt,descdef)+

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: dtxtatt:desctab --
>
]]><!-- desctxt -->

                      <!-- Descriptive Text -->
<![ %desctxt; [
<!element
   desctxt        -- Descriptive text --
                  -- Clause: 6.7.2.3 --
   O O
   (#PCDATA)      -- Lextype: words --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
]]><!-- desctxt -->

                 <!-- Descriptive Text Definition -->
<![ %desctxt; [
<!element
   descdef        -- Descriptive text definition --
                  -- Clause: 6.7.2.4 --
   O O
   (%HyCFC;)*

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
]]><!-- desctxt -->


<!entity % actypes "IGNORE">

                  <!-- User activity types -->
<![ %actypes; [
   <!entity %
      dactypes    -- Are default activity types fixed? --
                  -- Clause: 6.7.3 --

      ''          -- Activity types not fixed --
   >
   <!ENTITY % activity "INCLUDE" >
]]><!-- actypes -->
```

```
<!entity % activity "IGNORE">

              <!-- Activity Policy Common Attributes -->
<![ %activity; [
<!attlist
-- activity --    -- Activity policy association rules --
                  -- Clause: 6.7.3 --
   #ALL

   actrules       -- Activity policy association rules --
                  -- Activity policy association rules that apply to
                     this element. --
      CDATA       -- Reference --
                  -- Reftype: actrule* --
      #IMPLIED    -- Default: No activity policy association rule is
                     specified by this element as governing this
                     element. --
>
]]><!-- activity -->

              <!-- Activity policy association rule -->
<![ %activity; [
<!entity %
   dactypes       -- Are default activity types fixed? --
                  -- Clause: 6.7.3 --

   '#FIXED'       -- Activity types are fixed --
>
<!element
   actrule        -- Activity policy association rule --
                  -- Clause: 6.7.3 --
                  -- Optionally establishes association or
                     dissociation of activity policies with
                     information objects, and optionally nullifies
                     the effectivity of other actrules. --
   - O
   (%HyCFC;)*

-- Attributes [base]: actrule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: activity:actrules, actrule:nullify --
>
<!attlist
   actrule        -- Activity policy association rule --
                  -- Clause: 6.7.3 --

   governed       -- Governed objects --
                  -- Object(s) with which are associated or
                     dissociated the policies specified by the set of
                     activity type attributes of this actrule. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No objects are specified by this rule
                     as being governed by this rule. --
   associat       -- Associate or dissociate --
                  -- Specifying "assoc" creates associations, "dissoc"
                     dissolves associations created by other
                     actrules. --
      (assoc|dissoc)
      assoc

   nullify        -- Nullify activity policy association rules --
                  -- Nullify the effectivity of the actrules
                     referenced. --
      CDATA       -- Reference --
                  -- Reftype: actrule* --
      #IMPLIED

   actypes        -- Activity types --
                  -- List of the names of activity type attributes. --
                  -- Note: Any, all, or none of the members of the
                     HyTime default activity type set may be used, in
                     addition to any application-defined set. --
                  -- Constraint: Cannot redefine semantics of members
                     of the HyTime-defined default set. --
      NAMES       -- Lextype: ATTNAME+ --
      %dactypes;  -- Note: Will be #FIXED if actypes not supported --
      "access copy create delete link modify reschdul schdul trnscl
       unlink unschdul untrnscl"
                  -- Constant --

-- The following are the twelve HyTime-defined "activity type"
   attributes.  Use of these names always invokes their
   HyTime-defined semantics. --

   create         -- Object creation policies --
                  -- Policies that applied when any of the objects
                     were originally created. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object creation policies specified --

   copy           -- Object copying policies --
                  -- Policies that apply when attempts are made to
                     copy any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object copying policies specified --

   delete         -- Object deletion policies --
                  -- Policies that apply when attempts are made to
                     delete any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object deletion policies specified --

   access         -- Object access policies --
                  -- Policies that apply when attempts are made to
                     access any of the objects. --
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object access policies specified --
```

```
modify          -- Object modification policies --
                -- Policies that apply when attempts are made to
                   modify any of the objects --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object modification policies
                   specified --

link            -- Object hyperlinking policies --
                -- Policies that apply when attempts are made to
                   make any of the objects the anchor of a
                   hyperlink. --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object hyperlinking policies
                   specified --

unlink          -- Object unhyperlinking policies --
                -- Policies that apply when attempts are made to
                   make any of the objects no longer the anchor of a
                   hyperlink. --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object unhyperlinking policies
                   specified --

schdul          -- Object scheduling policies --
                -- Policies that apply when attempts are made to
                   schedule any of the objects in an FCS. --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object scheduling policies
                   specified --

unschdul        -- Object unscheduling policies --
                -- Policies that apply when attempts are made to
                   remove any given appearance of any of the objects
                   in an FCS. --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object unscheduling policies
                   specified --

reschdul        -- Object rescheduling policies --
                -- Policies that apply when attempts are made to
                   change the effective extent of any appearance of
                   any of the objects in an FCS. --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object rescheduling policies
                   specified --

trnscl          -- Object transclusion policies --
                -- Policies that apply when attempts are made to
                   transclude any of the objects. --
   CDATA        -- Reference --
   #IMPLIED     -- Default: No object transclusion policies
                   specified --

untrnscl        -- Object untransclusion policies --
                -- Policies that apply when attempts are made to
                   make any of the objects no longer transcluded. --
```

```
      CDATA       -- Reference --
      #IMPLIED    -- Default: No object untransclusion policies
                     specified --
>
]]><!-- activity -->

           <!-- HyTime Dimension Specification Notation -->
<![ %HyDimSpc; [
<!notation
   HyDimSpc       -- HyTime Dimension Specification Notation --
                  -- Clause: 6.8.2 --
                  -- A pair of axis markers (in the HyTime Axis Marker
                     List Notation, after resolution of marker
                     functions) that together specify a position and
                     quantum count along a single axis. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Dimension Specification Notation//EN"

-- Attributes [base]: HyDimSpc --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDimSpc       -- HyTime Dimension Specification Notation --
                  -- Clause: 6.8.2 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyMrkLst
>
<!entity % HyMrkLst "INCLUDE">
]]><!-- HyDimSpc -->


<!entity % HyDimLst "IGNORE">

               <!-- HyTime Dimension List Notation -->
<![ %HyDimLst; [
<!notation
   HyDimLst       -- HyTime Dimension List Notation --
                  -- Clause: 6.8.4 --
                  -- A list of dimensions each of which may be
                     specified as either a pair of axis markers (in
                     the HyTime Marker List Notation), or as all or
                     part of the list of axis markers returned by a
                     marker function. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Dimension List Notation//EN"

-- Attributes [base]: HyDimLst --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
```

```
<!attlist #NOTATION
   HyDimLst       -- HyTime Dimension List Notation --
                  -- Clause: 6.8.4 --

   GenArc   NAME     #FIXED GABridN
   superdcn NAME     #FIXED HyMrkLst
>
<!entity % HyMrkLst "INCLUDE">
]]><!-- HyDimLst -->


<!entity % HyMrkLst "IGNORE">

              <!-- HyTime Axis Marker List Notation -->
<![ %HyMrkLst; [
<!notation
   HyMrkLst       -- HyTime Axis Marker List Notation --
                  -- Clause: 6.8.1 --
                  -- A list of HyTime axis markers (signed non-zero
                     integers) and marker functions (elements that
                     conform to the markfun architectural form).
                     Marker functions are evaluated to lists of
                     markers. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Axis Marker List Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
]]><!-- HyMrkLst -->


<!entity % HyFunk "IGNORE">

               <!-- HyTime Marker Function Language -->
<![ %HyFunk; [
<!notation
   HyFunk         -- HyTime Marker Function Language --
                  -- Clause: 6.8.6 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Marker Function Language//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
]]><!-- HyFunk -->


<!entity % markfun "IGNORE">

                       <!-- Marker function -->
<![ %markfun; [
<![ %HyFunk; [
   <!entity % dmarkfun "HyFunk">
]]>
<!entity %
   dmarkfun       -- Default marker function notation --
                  -- Clause: 6.8.1.2 --

   "#REQUIRED"
>
<!element
   markfun        -- Marker function --
                  -- Clause: 6.8.1.2 --
                  -- Evaluates to a list of zero or more axis
                     markers. --
   O O
   (%HyCFC;)*     -- Constraint: content must conform to specified
                     marker function notation. --

-- Attributes [base]: markfun --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   markfun        -- Marker function --
                  -- Clause: 6.8.1.2 --

   notation       -- Marker function notation --
      NAME        -- Lextype: NOTATION --
      %dmarkfun;
>
]]><!-- markfun -->


<!entity % listloc "IGNORE">

                    <!-- List Location Address -->
<![ %listloc; [
<!element
   listloc        -- List location address --
                  -- Clause: 7.10.1.2 --
                  -- Locates nodes in a node list --
   - O
   (%dimlist;)*

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
```

```
<!entity % overrun "INCLUDE">
]]><!-- listloc -->


<!entity % pathloc "IGNORE">

                    <!-- Path Location Address -->
<![ %pathloc; [
<!element
   pathloc        -- Path location address --
                  -- Clause: 7.10.1.5 --
                  -- Locates nodes in a tree viewed as a path list --
   - O
   (%dimlist;)*   -- Constraint: resolved axis markers are interpreted
                     as a single list of pairs of dimension
                     specifications. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treecom,
   treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- pathloc -->


<!entity % relloc "IGNORE">

                  <!-- Relative Location Address -->
<![ %relloc; [
<!element
   relloc         -- Relative location address --
                  -- Clause: 7.10.1.6 --
                  -- Locates nodes in a tree relative to a starting
                     node --
   - O
   (%dimlist;)*   -- Constraint: resolved axis markers are interpreted
                     as a single list of dimension specifications. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc, relloc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   relloc         -- Relative location address --
                  -- Clause: 7.10.1.6 --
   strtnode       -- Starting node(s) --
                  -- Starting node(s) whose relatives are to be addressed --
      CDATA       -- Reference --
      #IMPLIED    -- Default: root(s) of location source tree(s) --

   nostart        -- No start --
                  -- Expected system behavior if no starting node is addressed
                     for location source. --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   relation       -- Relationship to starting node --
      (anc|children|esib|followng|parent|precedng|ysib)
      parent
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- relloc -->


<!entity % treeloc "IGNORE">

                    <!-- Tree Location Address -->
<![ %treeloc; [
<!element
   treeloc        -- Tree location address --
                  -- Clause: 7.10.1.4 --
                  -- Locates nodes in a tree by classical method --
   - O
   (%marklist;)*  -- Constraint: resolved axis markers are interpreted
                     as a single list of dimension specifications,
                     marker1 only; implied marker2 is 1. --

-- Attributes [base]: overrun --
-- Attributes [locs]: locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treecom,
   treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]><!-- treeloc -->


<!entity % calibrat "IGNORE">

                      <!-- Axis calibration -->
<![ %calibrat; [
<!attlist
-- calibrat --    -- Axis calibration --
```

```
                 -- Clause: 9.3.1 --
   (fcs)

   calibrat       -- Axis calibration --
                  -- The calibration of an axis is defined by
                     specifying the name of the axis followed by the
                     name of an attribute of the client element whose
                     value determines the position of the axis with
                     respect to an external context. --
      CDATA       -- Lextype: (AXISNM,ATTNAME,NOTATION)+ --
                  -- Constraint: Each axis name may appear only
                     once. --
                  -- Constraint: Calibration points should be
                     expressed in notations which make sense, given
                     the measurement domains of their corresponding
                     axes. --
      #IMPLIED    -- Default: no axis calibration --

-- Attributes named by the axis calibration attributes have an
   unnormalized lexical type of:

   (s*,granule?,s+,snzi,s+,("STARTS"|"ENDS") #ORDER SGMLCASE,s,char*)

   The first part of the attribute value (up to the last s separator)
   specifies the calibration point for the axis.  The rest of the
   value (that which matches char*) is a position within an external
   context, and is specified in the notation given following the
   attribute's name in the axis calibration attribute. --
>
<!entity % sched "INCLUDE">
]]><!-- calibrat -->


<!entity % HyCalSpc "IGNORE">

           <!-- HyTime Calendar Specification Notation -->
<![ %HyCalSpc; [
<!notation
   HyCalSpc       -- HyTime Calendar Specification Notation --
                  -- Clause: 9.9.1 --
                  -- Specifies a point in real time. The point
                     specified is that which occurs at the beginning
                     of the smallest unit of time in terms of which
                     the point is specified. --
                  -- Note: For example, the calendar specification
                     "1996-02-07" identifies the point occurring at
                     midnight between February 1st and February 2nd of
                     1996. --
                  -- Note: Can be used directly for axis calibration
                     and alignment.  Also used as the basis for a
                     marker function notation; see HyCalFun. --
                  -- Lextype: ((JULDATE|fulldate|monthday|day)?,
                               (UTCtime|hrminsec|hrmin|minute)?,
                              timeoff*) --
                  -- Constraint: Date must be valid (e.g. not
                     "2-30"). --
                  -- Constraint: Valid time followed by list of
                     applicable non-conflicting time offsets for
                     daylight-saving or time zones. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Calendar Specification Notation//EN"

-- Attributes [sched]: HyCalSpc --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyCalSpc       -- HyTime Calendar Specification Notation --
                  -- Clause: 9.9.1 --

   GMTorUTC       -- GMT or UTC timescale --
                  -- Note: (GMT noon = UTC midnight) --
      (GMT|UTC)
      UTC
>
<!entity % sched "INCLUDE">
]]><!-- HyCalSpc -->


<!entity % grpdex "IGNORE">

             <!-- Group Derived Extent Specification -->
<![ %grpdex; [
<!attlist
-- grpdex --      -- Group derived extent specification --
                  -- Clause: 9.4.3 --
   (evgrp,modgrp,progrp)

   grpdex         -- Group derived extent specification --
                  -- Use for resizing, rearrangement, repetition.
                     Group members are given the derived extents.  If
                     nominal extent is also wanted, it must be
                     specified as one of the derived extents. --
      CDATA       -- Reference --
                  -- Reftype: (extent|extlist)* --
      #IMPLIED    -- Default: grpdex is same as overall "group scope",
                     that is, the nominal extents are effective --
>
<!entity % sched "INCLUDE">
]]><!-- grpdex -->


<!entity % grprepet "IGNORE">

               <!-- Group Repetition Specification -->
<![ %grprepet; [
<!attlist
-- grprepet --    -- Group repetition specification --
                  -- Clause: 9.4.3 --
   (evgrp,modgrp,progrp)
   repscope       -- Repetition scope --
                  -- Scope within which the contained events,
                     modscopes or proscopes are repeated, subject to
                     additional criteria in the value of each
                     axis-named attribute, if any.  For each axis,
                     repetition starts at the first quantum of the
                     repetition scope and continues until the
                     repetition scope is exhausted on each axis. --
       CDATA      -- Reference --
                  -- Reftype: extent --
       #IMPLIED   -- Default: The first or only occurrence of the
                     group is wherever the effective extent of the
                     group places it, and the number of repetitions is
                     limited only by the repeat count limit parameters
                     of the axis-named attributes. --

-- Additional attributes may be defined for client elements, one for
   each axis of the FCS.  The names of the attributes are the axis
   names, and the values of the attributes are specifications
   regarding repetitions of the group and the implicit dimref of the
   succeeding extent along an axis.  Each of the attributes has the
   lexical model (uint,(("NOPART"|"PART"),unzi)?,unzi?), which is
   interpreted as follows:

      "repeat count limit" parameter (uint):
         If the value is 0, there is no repeat count limit.  The group
         is repeated on this axis from the beginning to the end of the
         dimension specified by the repscope attribute.  If no value
         is specified for the repscope attribute, the 0 value is
         treated as if it were 1.  If the value of the "repeat count
         limit" parameter is greater than 0, the number of repetitions
         is limited to the number specified.  If no value is specified
         for the repscope attribute, the number of repetitions is
         limited only by the number of repetitions specified and the
         remaining length of the axis.  If no value is specified for
         an axis-named attribute, the "repeat count limit" parameter
         is treated as if it were 1.

      "allow partial dimensions" parameter ("PART"|"NOPART"):
         If the value is "PART", partial repetitions at the beginning
         and/or end of the series on this axis will be used to
         completely fill the repetition scope.  If the value is
         "NOPART", partial repetitions do not occur.  (There may be
         unoccupied space at the beginning and/or end of the series on
         this axis.)  The default value is "NOPART".

      "start point within pattern" parameter (penultimate unzi):
         The value is the number of the HMU (where the "first"
         component of the group's dimension on this axis is 1) at
         which the first occurrence in the series begins.  If the
         value is not 1, the remaining partial portion of the group
         will constitute the first repetition in each series.  This
         first partial repetition will occur as skipped space if the
         "allow partial dimensions" parameter's value is "NOPART".
         The default value is 1.
      "offset to next occurrence" parameter (final unzi):
         The value is the number of HMUs that will be treated as the
         qcnt of the group for purposes of determining the location of
         the next contiguous repetition (or the next dimension with an
         implicit dimref used as its "first" component).  The default
         value is the effective qcnt of the group. --
>
<!entity % sched "INCLUDE">
]]><!-- grprepet -->


<!entity % objalign "IGNORE">

                    <!-- Object alignment rule -->
<![ %objalign; [
<!attlist
-- objalign --    -- Object alignment rule --
                  -- Clause: 9.5.1 --
   (event)

   align          -- Object alignment rule --
                  -- The alignment of an object with an event on each
                     axis can be defined by specifying the name of the
                     axis followed by one of the keywords #LEFT,
                     #RIGHT, or #CENTER.  Alternatively, the name of
                     any axis may be followed by the name of an
                     attribute of the client element whose value
                     determines the position of the object with
                     respect to the event. --
      CDATA       -- Lextype: (AXISNM,(("#LEFT"|"#RIGHT"|#CENTER)|
                                  (ATTNAME,NOTATION)))+ --
                  -- Constraint: Each axis name may appear only
                     once. --
      #IMPLIED    -- Default: no explicit alignment --

-- Attributes named by the object alignment rule attribute have
   an unnormalized lexical type of:

   (s*,granule?,s+,snzi,s+,("STARTS"|"ENDS") #ORDER SGMLCASE,s,char*)

   The first part of the attribute value (up to the last s separator)
   specifies the calibration point within the event.  The rest of the
   value (that which matches char*) is a position within the object,
   and is specified in the notation given following the attribute's
   name in the object alignment attribute. --
>
<!entity % sched "INCLUDE">
]]><!-- objalign -->


<!entity % pulsemap "IGNORE">

                          <!-- Pulsemaps -->
<![ %pulsemap; [
<!attlist
```

```
-- pulsemap --    -- Pulse Maps --
                  -- Clause: 9.7 --
   (baton,event,evgrp,evsched,modgrp,modscope,progrp,proscope,wand)

   pulsemap       -- Pulse Maps --
                  -- Schedules to be used as pulse maps. --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)* --
                  -- Constraint: Pulse maps must be in same FCS as
                     referrer. --
      #IMPLIED    -- Default: none --
>
<!entity % sched "INCLUDE">
]]><!-- pulsemap -->


<!entity % wndpatch "IGNORE">

                         <!-- Wand Patch -->
<![ %wndpatch; [
<!element
   wndpatch       -- Wand patch --
                  -- Clause: 10.2.4 --
   - O
   (#PCDATA)      -- Reference --
                  -- Reftype: (wndpatch|wand)* --
                  -- Constraint: subset of SGML model group with
                     IDREFs as names and no occurrence indicators or
                     OR connectors. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: wandrule:wand, wndpatch --
>
<!entity % wand "INCLUDE">
]]><!-- wndpatch -->


<!entity % wand "IGNORE">

                          <!-- Wand Rule -->
<![ %wand; [
<!element
   wandrule       -- Wand rule --
                  -- Clause: 10.2.3.1 --
   - O
   (%HyCFC;)*

-- Attributes [rend]: wandrule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
```

```
>
<!attlist
   wandrule       -- Wand rule --
                  -- Clause: 10.2.3.1 --

   evscheds       -- Event schedules --
      CDATA       -- Reference --
                  -- Reftype: (evsched|wand)* --
                  -- Constraint: all evscheds and wands must be in
                     same logical FCS as all of the wands referenced
                     by the wand attribute. --
      #REQUIRED

   wand           -- Wand or wand patch --
      CDATA       -- Reference --
                  -- Reftype: (wand|wndpatch) --
      #IMPLIED    -- Default: none --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->

                            <!-- Wand -->
<![ %wand; [
<!element
   wand           -- Wand --
                  -- Clause: 10.2.3.2 --
   - O
   (modgrp|modscope)+

-- Attributes [sched]: sched --
-- Attributes [rend]: select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, dimref:schdspec,
   pdimref:prjtarg, pulsemap:pulsemap, rfcsloc:prjsrc --
-- Referrers [rend]: batrule:scheds, batrule:targschd,
   prorule:sources, prorule:targschd, wandrule:wand, wndpatch --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->

                       <!-- Modifier Scope -->
<![ %wand; [
<!element
   modscope       -- Modifier scope --
                  -- Defines the scope of a modifier as one or more
                     extents. --
                  -- Clause: 10.2.3.3 --
   - O
   (%HyCFC;)*

-- Attributes [base]: overrun --
-- Attributes [sched]: exspec --
-- Attributes [rend]: modscope, select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!attlist
   modscope       -- Modifier scope --
                  -- Clause: 10.2.3.3 --

   modifier       -- Modifiers and modifier patches to be
                     applied --
      CDATA       -- Reference --
      #IMPLIED    -- Default: the modification semantics, if any, are
                     specified directly by the GI, attributes and/or
                     content of this element in some
                     application-defined fashion --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->

                    <!-- Modifier Scope Group -->
<![ %wand; [
<!element
   modgrp         -- Modifier scope group --
                  -- Clause: 10.2.3.4 --
   - O
   (modgrp|modscope)+

-- Attributes [rend]: select --
-- OptionalAttributes [sched]: grpdex, grprepet, pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!entity % modify "INCLUDE">
]]><!-- wand -->


<!entity % patch "IGNORE">

                       <!-- Modifier Patch -->
<![ %patch; [
<!element
   modpatch       -- Modifier patch --
                  -- Clause: 10.2.4 --
   - O
   (#PCDATA)      -- Reference --
                  -- Note: IDREFs may refer to modpatch
                     elements. --
```

```
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: modpatch, modrule:modpatch --
>
<!entity % modify "INCLUDE">
]]><!-- patch -->


<!entity % modify "IGNORE">

                        <!-- Modifier Rule -->
<![ %modify; [
<!element
   modrule        -- Modifier rule --
                  -- Clause: 10.2.2 --
   - O
   (%HyCFC;)*

-- Attributes [rend]: modrule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!attlist
   modrule        -- Modifier rule --
                  -- Clause: 10.2.2 --

   events         -- Events and modscopes --
      CDATA       -- Reference --
                  -- Reftype:
                     (event|evgrp|evsched|modscope|modgrp|wand)* --
      #REQUIRED

   modifier       -- Modifier or modifier patch --
      CDATA       -- Reference --
                  -- Constraint: only one modifier or modifier
                     patch --
      #REQUIRED
>
<!entity % rend "INCLUDE">
]]><!-- modify -->


<!entity % HyPro "IGNORE">

                  <!-- HyTime Projector Notation -->
<![ %HyPro; [
<![ %HyExtLst; [
   <!entity % dpextlst "HyExtLst">
]]>
<!entity %
```

```
   dpextlst       -- Default projector extent list notation --
                  -- Clause: 10.3.1.1 --

   "#IMPLIED"
>
<!notation
   HyPro          -- HyTime Projector Notation --
                  -- Clause: 10.3.1.1 --
                  -- Defines projection functions by deriving linear
                     relationships between each dimension of each of
                     the specified extents (in the target FCS) and its
                     corresponding dimension in the source FCS.
                     Projector functions for individual extents or
                     dimensions may also be specified using expro or
                     dimpro elements. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Projector Notation//EN"

-- Attributes [rend]: HyPro --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyPro          -- HyTime Projector Notation --
                  -- Clause: 10.3.1.1 --

   extlistn       -- Extent list notation --
                  -- Notation used to specify the target extents. --
      NAME        -- Lextype: NOTATION --
      %dpextlst;  -- Default: Only expro, dimpro, extent, and extlist
                     elements allowed. --
>
<!entity % project "INCLUDE">
]]><!-- HyPro -->


<!entity % HyExPro "IGNORE">

              <!-- HyTime Extent Projector Notation -->
<![ %HyExPro; [
<![ %HyExSpec; [
   <!entity % dpexspec "HyExSpec">
]]>
<!entity %
   dpexspec       -- Default projector extent specification
                     notation --
                  -- Clause: 10.3.1.3 --

   "#IMPLIED"
>
<!notation
   HyExPro        -- HyTime Extent Projector Notation --
                  -- Clause: 10.3.1.3 --
                  -- Defines projection functions by deriving linear
```

```
                     relationships between each dimension of the
                     specified extent (in the target FCS) and its
                     corresponding dimension in the source FCS.
                     Projector functions for particular dimensions may
                     also be specified using dimension projector
                     elements. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Extent Projector Notation//EN"

-- Attributes [rend]: HyExPro --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyExPro        -- HyTime Extent Projector Notation --
                  -- Clause: 10.3.1.3 --

   exspnot        -- Extent specification notation --
                  -- Notation used to specify the target extent. --
      NAME        -- Lextype: NOTATION --
      %dpexspec;  -- Default: Only dimpro and dimspec elements
                     allowed. --
>
<!entity % project "INCLUDE">
]]><!-- HyExPro -->


<!entity % HyDimPro "IGNORE">

             <!-- HyTime Dimension Projector Notation -->
<![ %HyDimPro; [
<![ %HyDimSpc; [
   <!entity % dpdimspc "HyDimSpc">
]]>
<!entity %
   dpdimspc       -- Default projector dimension specification
                     notation --
                  -- Clause: 10.3.1.5 --

   "#REQUIRED"
>
<!notation
   HyDimPro       -- HyTime Dimension Projector Notation --
                  -- Clause: 10.3.1.5 --
                  -- Defines a projection function by deriving a
                     linear relationship between the specified
                     dimension (in the target FCS) and its
                     corresponding dimension in the source FCS. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Extent Projector Notation//EN"

-- Attributes [rend]: HyDimPro --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDimPro       -- HyTime Dimension Projector Notation --
                  -- Clause: 10.3.1.5 --

   dimspnot       -- Dimension specification notation --
                  -- Notation used to specify the target dimension. --
      NAME        -- Lextype: NOTATION --
      %dpdimspc;
>
<!entity % project "INCLUDE">
]]><!-- HyDimPro -->


<!entity % proseq "IGNORE">

                     <!-- Projector sequence -->
<![ %proseq; [
<!element
   proseq         -- Projector sequence --
                  -- Clause: 10.3.2.3 --
   - O
   (#PCDATA)      -- Reference --
                  -- Reftype: (proseq|projectr)* --
                  -- Constraint: subset of SGML model group with
                     IDREFs as names and no occurrence indicators, OR
                     connectors, or AND connectors. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: prorule:proseq, proseq --
>
<!entity % project "INCLUDE">
]]><!-- proseq -->


<!entity % batonseq "IGNORE">

                       <!-- Baton sequences -->
<![ %batonseq; [
<!element
   batonseq       -- Baton sequence --
                  -- Clause: 10.3.3.5 --
   - O
   (#PCDATA)      -- Reference --
                  -- Reftype: (batonseq|baton)* --
                  -- Constraint: subset of SGML model group with
                     IDREFs as names and no occurrence indicators, OR
                     connectors, or AND connectors. --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
```

```
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: batonseq, batrule:baton --
>
<!entity % baton "INCLUDE">
]]><!-- batonseq -->


<!entity % baton "IGNORE">

                         <!-- Baton Rule -->
<![ %baton; [
<!element
   batrule        -- Baton rule --
                  -- Clause: 10.3.3.1 --
   - O
   (%HyCFC;)*

-- Attributes [rend]: batrule, modified --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!attlist
   batrule        -- Baton rule --
                  -- Clause: 10.3.3.1 --

   scheds         -- Schedules --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)* --
      #REQUIRED

   baton          -- Baton --
      CDATA       -- Reference --
                  -- Reftype: (baton|batonseq) --
      #REQUIRED

   targschd       -- Target schedules --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|wand)+ --
      #REQUIRED
>
<!entity % project "INCLUDE">
]]><!-- baton -->

                            <!-- Baton -->
<![ %baton; [
<!element
   baton          -- Baton --
                  -- Clause: 10.3.3.2 --
   - O
   (progrp|proscope)+

-- Attributes [sched]: sched --
-- Attributes [rend]: select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, dimref:schdspec, pdimref:prjby,
   pdimref:prjtarg, pulsemap:pulsemap, rfcsloc:prjby,
   rfcsloc:prjsrc --
-- Referrers [rend]: batonseq, batrule:baton, batrule:scheds,
   batrule:targschd, prorule:sources, prorule:targschd --
>
<!entity % project "INCLUDE">
]]><!-- baton -->

                       <!-- Projector Scope -->
<![ %baton; [
<!element
   proscope       -- Projector scope --
                  -- Clause: 10.3.3.3 --
                  -- Defines the scope of a projector as one or more
                     extents. --
   - O
   (%HyCFC;)*

-- Attributes [base]: overrun --
-- Attributes [sched]: exspec --
-- Attributes [rend]: proscope, select --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, pdimref:prjby, rfcsloc:prjby,
   rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!attlist
   proscope       -- Projector scope --
                  -- Clause: 10.3.3.3 --

   projectr       -- Scheduled projectors --
      CDATA       -- Reference --
                  -- Reftype: projectr* --
      #IMPLIED    -- Default: none --
>
<!entity % project "INCLUDE">
]]><!-- baton -->

                    <!-- Projector Scope Group -->
<![ %baton; [
<!element
   progrp         -- Projector scope group --
                  -- Clause: 10.3.3.4 --
   - O
   (progrp|proscope)+
-- Attributes [rend]: select --
-- OptionalAttributes [sched]: grpdex, grprepet, pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, pdimref:prjby, rfcsloc:prjby,
   rfcsloc:prjsrc --
-- Referrers [rend]: prorule:sources --
>
<!entity % project "INCLUDE">
]]><!-- baton -->


<!entity % project "IGNORE">

                          <!-- Projector -->
<![ %project; [
<![ %HyPro; [
   <!entity % dpro "HyPro">
]]>
<!entity %
   dpro           -- Default projector notation --
                  -- Clause: 10.3.1 --

   "#IMPLIED"
>
<!element
   projectr       -- Projector --
                  -- Clause: 10.3.1 --
                  -- Defines a projection function for events,
                     modscopes, and proscopes in a source FCS. --
   O O
   (%HyCFC;|%dimlist;|dimpro|expro|extent|extlist)*
                  -- Constraint: If no extent projector function
                     notation is specified, content is restricted to
                     (expro)* --

-- Attributes [rend]: projectr --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: prorule:proseq, proscope:projectr, proseq --
>
<!attlist
   projectr       -- Projector --
                  -- Clause: 10.3.1 --

   notation       -- Projector notation --
      NAME        -- Lextype: NOTATION --
      %dpro;      -- Default: Content restricted to (expro)* --

   strict         -- Strictness --
                  -- Indication of the strictness with which the
```

```
                     projection function is intended to be
                     executed. --
      CDATA
      #IMPLIED    -- Default: No indication --
>
<!entity % rend "INCLUDE">
]]><!-- project -->

                      <!-- Extent Projector -->
<![ %project; [
<![ %HyExPro; [
   <!entity % dexpro "HyExPro">
]]>
<!entity %
   dexpro         -- Default extent projector notation --
                  -- Clause: 10.3.1.2 --

   "#IMPLIED"
>
<!element
   expro          -- Extent Projector --
                  -- Clause: 10.3.1.2 --
                  -- Defines projection functions to a single target
                     extent for extents in a source FCS. --
   O O
   (%HyCFC;|%dimlist;|dimpro)*
                  -- Constraint: If no extent projector notation is
                     specified, content is restricted to (dimpro)* --

-- Attributes [rend]: expro --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   expro          -- Extent Projector --
                  -- Clause: 10.3.1.2 --

   notation       -- Extent projector notation --
      NAME        -- Lextype: NOTATION --
      %dexpro;    -- Default: Content restricted to (dimpro)* --
>
<!entity % rend "INCLUDE">
]]><!-- project -->

                     <!-- Dimension Projector -->
<![ %project; [
<![ %HyDimPro; [
   <!entity % ddimpro "HyDimPro">
]]>
<!entity %
   ddimpro        -- Default dimension projector notation --
                  -- Clause: 10.3.1.4 --

   "#REQUIRED"
```

```
>
<!element
   dimpro         -- Dimension projector --
                  -- Clause: 10.3.1.4 --
                  -- Defines a projection function for dimensions
                     along an axis of a source FCS. --
   O O
   (%HyCFC;|%marklist;)*

-- Attributes [rend]: dimpro --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dimpro         -- Dimension projector --
                  -- Clause: 10.3.1.4 --

   notation       -- Dimension projector notation --
      NAME        -- Lextype: NOTATION --
      %ddimpro;
>
<!entity % rend "INCLUDE">
]]><!-- project -->

                       <!-- Projector Rule -->
<![ %project; [
<!element
   prorule        -- Projector rule --
                  -- Clause: 10.3.2.2 --
                  -- Direct association between sources, projectors,
                     and target schedules. --
   - O
   (%HyCFC;)*

-- Attributes [rend]: modified, prorule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: pdimref:prjby, rfcsloc:prjby --
-- Referrers [rend]: rendrule --
>
<!attlist
   prorule        -- Projector rule --
                  -- Clause: 10.3.2.2 --

   sources        -- Projection sources --
                  -- Sources for projection --
      CDATA       -- Reference --
                  -- Reftype: (baton|event|evgrp|evsched|modgrp|
                               modscope|progrp|proscope|wand)* --
      #REQUIRED

   projectr       -- Projectors and projector sequences --
```

```
                  -- Projectors and/or projector sequences to be used
                     for projection from sources to targets. --
      CDATA       -- Reference --
                  -- Reftype: (projectr|proseq)* --
      #REQUIRED

   targschd       -- Target schedules --
                  -- Schedules onto which sources will be projected by
                     projectors and/or projector sequences. --
      CDATA       -- Reference --
                  -- Reftype: (evsched|wand|baton)+ --
                  -- Constraint: Only events will be projected onto
                     event schedule targets; only modscopes will be
                     projected onto wands, and only proscopes will be
                     projected onto batons. --
      #REQUIRED
>
<!entity % rend "INCLUDE">
]]><!-- project -->


<!entity % rend "IGNORE">

                 <!-- Precision of selection -->
<![ %rend; [
<!attlist
-- select --      -- Precision of selection --
                  -- Clause: 10.1.1 --
   (baton,modgrp,modscope,progrp,proscope,wand)

   select         -- Precision of selection --
                  -- How much of the event, modscope or proscope must
                     lie within the extent of the modifying modscope
                     or projecting proscope in order for modification
                     or projection to occur. --
      CDATA       -- Lextype: ("#ALL"|"#ANY"|(unzi,granule?))+ --
                  -- Constraint: One specification for each axis or
                     one for all axes. --
                  -- Constraint: Precision unzi is specified in terms
                     of the following granule, if supplied; otherwise
                     in terms of the axis HMU defined for the schedule. --
      #IMPLIED    -- Default: when not specified on a wand or baton, #ALL.
                     When not specified on a modgrp, modscope, progrp, or
                     proscope inherits effective value of containing wand,
                     modgrp, baton or progrp. --

   portion        -- Portion modified or projected --
                  -- Modification or projection applies to entire
                     event, modscope or proscope (whole), or only to
                     the portions that are co-located with the
                     modifying modscope or projecting proscope
                     (part). --
      NAMES       -- Lextype: ("WHOLE"|"PART")+ --
                  -- Constraint: One specification for each axis or
                     one for all axes. --
      #IMPLIED    -- Default: when not specified on a wand or baton,
                     WHOLE. When not specified on a modgrp,
                     modscope,progrp, or proscope inherits effective
                     value of containing baton,modgrp, progrp, or
                     wand. --
>
<!entity % sched "INCLUDE">
]]><!-- rend -->

                       <!-- Rendition Rule -->
<![ %rend; [
<!element
   rendrule       -- Rendition rule --
                  -- Clause: 10.4 --
   - O
   (%ArcCFC;)*    -- Reference --
                  -- Reftype:
                     (batrule|modrule|prorule|rendrule|wandrule) --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [rend]: rendrule --
>
<!entity % sched "INCLUDE">
]]><!-- rend -->


<!entity % sched "IGNORE">

                            <!-- Event -->
<![ %sched; [
<!element
   event          -- Scheduled event --
                  -- Clause: 9.5.1 --
   - O
   (%HyCFC;)*

-- Attributes [base]: overrun --
-- Attributes [sched]: exspec --
-- OptionalAttributes [sched]: pulsemap, objalign --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: modrule:events, prorule:sources --
>
<!attlist
   event          -- Event --
                  -- Clause: 9.5.1 --

   object         -- Event objects --
      CDATA       -- Reference --
      #IMPLIED    -- Default: Event is the object --
>
```

```
<!entity % overrun "INCLUDE">
]]><!-- sched -->

                         <!-- Event Group -->
<![ %sched; [
<!element
   evgrp          -- Event group --
                  -- Clause: 9.5.2 --
   - O
   (event|evgrp)+

-- Attributes [base]: overrun --
-- OptionalAttributes [sched]: pulsemap, grpdex, grprepet --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, rfcsloc:prjsrc --
-- Referrers [rend]: modrule:events, prorule:sources --
>
]]><!-- sched -->


<!entity % overrun "IGNORE">

                      <!-- Overrun Handling -->
<![ %overrun; [
<!attlist
-- overrun --     -- Overrun handling --
                  -- Clause: 6.8.5 --
   (dataloc,dimref,event,evgrp,fcsloc,listloc,modgrp,modscope,
    pathloc,progrp,proscope,relloc,treeloc)

   overrun        -- Overrun Handling --
                  -- Handling of dimension that overruns range --
      (error|ignore|trunc|wrap)
      error
>
]]><!-- overrun -->


<!entity % grovplan "IGNORE">

                         <!-- Grove Plan -->
<![ %grovplan; [
<!element
   grovplan       -- Grove plan --
                  -- Clause: 7.1.4.1 --
                  -- Specifies a subset of a complete grove's address
                     space. --
   - O
   (inclmod?,omitmod?,inclclas?,omitclas?,inclprop?,omitprop?)
                  -- Constraint: Can include classes and properties
                     from omitted modules (but not properties from
                     omitted classes). --
```

```
-- Attributes [locs]: grovplan --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [locs]: agrovdef:grovplan, dgrvplan:grovplan,
   egrovplan:grovplan, lgrvplan:grovplan, pgrovdef:grovplan --
>
<!attlist
   grovplan       -- Grove plan --
                  -- Clause: 7.1.4.1 --

   propset        -- Property set definition document --
      ENTITY      -- Constraint: not an error if the property
                     set document entity is not available. --
      #REQUIRED
>
<!element
-- inomcomp --    -- Include component/omit component elements --
                  -- Clause: 7.1.4.1 --
   (inclclas,inclmod,omitclas,omitmod)

   - O
   (#PCDATA)      -- Lextype: cnmlist --

-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!element
-- inomprop --    -- Include property/omit property --
                  -- Clause: 7.1.4.1 --
   (inclprop,omitprop)

   - O
   (#PCDATA)      -- Lextype: cnmlist --

-- Attributes [locs]: inomprop --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
-- inomprop --    -- Include property/omit property --
                  -- Clause: 7.1.4.1 --
   (inclprop,omitprop)

   classes        -- Classes from which to include or omit the named
                     properties --
      NAMES
      #IMPLIED    -- Default: all classes that exhibit the
                     property. --
>
]]><!-- grovplan -->
```

```
                 <!-- HyTime Document Grove Plan -->
<![ %grovplan; [
<!attlist
-- dgrvplan --    -- HyTime document grove plan --
                  -- Clause: 7.1.4.1 --
   (HyDoc)

   grovplan       -- Grove plan --
                  -- Grove plan for HyTime extended SGML document
                     grove --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: HyTime default grove plan --
>
]]><!-- grovplan -->

                      <!-- Entity Grove Plan -->
<![ %grovplan; [
<!attlist #NOTATION
-- egrvplan --    -- Entity grove plan --
                  -- Clause: 7.1.4.1 --
   #ALL

   grovplan       -- Grove plan --
                  -- Grove plan to use when constructing grove from
                     entity. --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: default grove plan for entity's DCN. --
>
]]><!-- grovplan -->

<![ %grovplan; [
<!attlist
-- lgrvplan --    -- Location source grove plan --
                  -- Clause: 7.1.4.1 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmquery,nmsploc,
    pathloc,proploc,queryloc,relloc,treeloc)

    grovplan      -- Grove plans --
                  -- Grove plans that govern resolution of address --
       CDATA      -- Reference --
                  -- Reftype: grovplan* --
       #IMPLIED   -- Default: grove plans of location source groves as
                     defined by their grove definitions, whether
                     explicitly specified or implied. --
>
]]><!-- grovplan -->


<!entity % pgrovdef "IGNORE">

                  <!-- Primary Grove Definition -->
<![ %pgrovdef; [
<!element
```

```
   pgrovdef       -- Primary grove definition --
                  -- Clause: 7.1.4.4 --
   - O
   (%HyCFC;)*

-- Attributes [locs]: pgrovdef --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   pgrovdef       -- Primary grove definition --
                  -- Clause: 7.1.4.4 --

   grovesrc       -- Grove source --
      ENTITY
      #REQUIRED

   grovecon       -- Grove construction process --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: inherent in notation of grove source --

   grovplan       -- Grove plan --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: default grove plan of grove
                     construction process. --
>
]]><!-- pgrovdef -->


<!entity % agrovdef "IGNORE">

                 <!-- Auxiliary Grove Definition -->
<![ %agrovdef; [
<!element
   agrovdef       -- Auxiliary grove definition --
                  -- Clause: 7.1.4.4 --
   - O
   (%HyCFC;)*

-- Attributes [locs]: agrovdef --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   agrovdef       -- Auxiliary grove definition --
                  -- Clause: 7.1.4.4 --

   grovesrc       -- Grove source --
      CDATA       -- Reference --
      #REQUIRED
   grovecon       -- Grove construction process --
      NAME        -- Lextype: NOTATION --
      #REQUIRED

   grovplan       -- Grove plan --
      CDATA       -- Reference --
                  -- Reftype: grovplan --
      #IMPLIED    -- Default: default grove plan of grove
                     construction process. --
>
]]><!-- agrovdef -->


<!entity % nmsploc "IGNORE">

                 <!-- Name-Space Location Address -->
<![ %nmsploc; [
<!element
   nmsploc        -- Name-space location address --
                  -- Clause: 7.9.3 --
                  -- Addresses objects by name in a name-space --
                  -- Constraint: location source must be an object that
                     exhibits a named node list property or whose
                     additional property source exhibits a named node
                     list property. --
   - O
   (#PCDATA)      -- Lextype: (word|literal)* --

-- Attributes [locs]: nmsploc, proplat, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   nmsploc        -- Name-space location address --
                  -- Clause: 7.9.3 --

   namespc        -- Name-space --
                  -- Name-space property of location source from which
                     nodes are selected. --
      NAME
      #REQUIRED

   notspace       -- If nmspace name is invalid? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   notname        -- If name is not valid in nmspace? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
<!entity % proplat "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
```

```
]]><!-- nmsploc -->


<!entity % proploc "IGNORE">

                  <!-- Property Location Address -->
<![ %proploc; [
<!element
   proploc        -- Property location address --
                  -- Clause: 7.9.2 --
                  -- Locates property of a grove node --
   - O
   (#PCDATA)      -- Lextype: compname --

-- Attributes [locs]: proplat, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!entity % proplat "INCLUDE">
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- proploc -->

                   <!-- Query Location Address -->
<![ %queryloc; [
<!element
   queryloc       -- Query location address --
                  -- Clause: 7.11.1 --
                  -- Locates objects in a grove using queries against
                     their properties as defined in the property set
                     used to construct the grove. --
   - O
   (%HyCFC;)*

-- Attributes [locs]: locsrc, queryloc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   queryloc       -- Query location address --
                  -- Clause: 7.11.1 --

   notation       -- Query notation --
      NAME        -- Lextype: NOTATION --
      #REQUIRED

   notfound       -- No data found --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
```

```
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
]]><!-- queryloc -->


<!entity % locsrc "IGNORE">

                       <!-- Location Source -->
<![ %locsrc; [
<!attlist
-- locsrc --      -- Location source attributes --
                  -- Clause: 7.2 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmquery,nmsploc,
    pathloc,proploc,queryloc,relloc,treeloc)

   locsrc         -- location source --
      CDATA       -- Reference --
                  -- Constraint: References to entities are references
                     to roots of primary groves constructed from the
                     entities. --
      #IMPLIED

   cantcnst       -- Cannot construct grove from located data --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR
>
]]><!-- locsrc -->


<!entity % impsrc "IGNORE">

            <!-- Implied Location Source -->
<![ %impsrc; [
<!attlist
-- impsrc --      -- Implied location source --
                  -- Clause: 7.3 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmsploc,nmquery,
    pathloc,proploc,queryloc,relloc,treeloc)

   impsrc         -- Implied location source --
                  -- Default location source for implied locsrc
                     attribute.  Possible values are:

                     ptreert   Default location source is node in
                               default grove that serves as principal
                               tree root, normally the document
                               element.

                     grovert   Grove root of document that contains
                               location address.

                     referrer  Default location source is non-location
                               address object making direct or
                               indirect reference to the location
                               ladder of which this location address
                               is the top rung.
                     referatt  Default location source is attribute
                               of referrer named by referatt
                               attribute.
                     implicit  Location source is implicit in query.
                               For queryloc, location source is
                               implicit in query.  For other location
                               address forms, same as ptreert.
                     --
      (grovert|implicit|ptreert|referatt|referrer)
                  -- Constraint: referatt allowed only when referatt
                     option is supported. --
      ptreert
>
]]><!-- impsrc -->

    <!-- Attribute Of Referrer That Addresses Location Source -->
<![ %referatt; [
<!attlist
-- referatt --    -- Attribute of referrer that addresses location
                     source --
                  -- Clause: 7.3 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,nmlist,nmquery,nmsploc,
    pathloc,proploc,queryloc,relloc,treeloc)

   referatt       -- Attribute of referrer that addresses location
                     source --
      CDATA       -- Lextype: ATTORCON --
      #IMPLIED    -- Constraint: named attribute or content must be
                     exhibited by referrer and must be referential. --
>
]]><!-- referatt -->

                <!-- Multiple Location Attributes -->
<![ %multloc; [
<!attlist
-- multloc --     -- Multiple location attributes --
                  -- Clause: 7.4 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,mixedloc,nameloc,nmlist,
    nmquery,nmsploc,pathloc,proploc,queryloc,relloc,treeloc)

   ordering       -- Is ordering of locations significant? --
      (noorder|ordered)
      ordered

   set            -- Make multiple a set by ignoring duplicates --
      (notset|set)
      notset
>
]]><!-- multloc -->

                     <!-- Tree type attribute -->
<![ %treetype; [
<![ %spanloc; [
   <!entity % treetpel
      "anchloc,dataloc,fcsloc,linkloc,listloc,mixedloc,nameloc,nmlist,
       nmquery,nmsploc,pathloc,proploc,queryloc,relloc,treeloc"
   >
]]>
<!entity %
   treetpel       -- Tree type associated element forms --
                  -- Clause: 7.5 --

   "pathloc,relloc,treeloc"
>
<!attlist
-- treetype --    -- Tree type --
                  -- Clause: 7.5 --
   (%treetpel;)

   treetype       -- Tree type --
                  -- Type of tree to address as location source --
      (content|subnode)
      content
>
]]><!-- treetype -->

                        <!-- Span Location -->
<![ %spanloc; [
<!attlist
-- spanloc --     -- Span location attributes --
                  -- Clause: 7.6 --
   (anchloc,dataloc,fcsloc,linkloc,listloc,mixedloc,nameloc,nmlist,
    nmquery,nmsploc,pathloc,proploc,queryloc,relloc,treeloc)

   spanloc        -- Span location --
                  -- Do multiple locations comprise a span? --
      (nspan|spanloc)
      nspan

   limsort        -- Span limits sort --
                  -- Are limits sorted with span start first? --
      (limsort|nlimsort)
      nlimsort
>
]]><!-- spanloc -->


<!entity % reftype "IGNORE">

                  <!-- Reference Element Type -->
<![ %reftype; [
<!attlist
-- reftype --     -- Reference type attributes --
                  -- Clause: 7.7.1 --
   #ALL

   reftype        -- Reference element type --
      CDATA       -- Lextype: (("#ALL",(GI|modelgroup|"#ANY"))?,
                               (ATTORCON,(GI|modelgroup|"#ANY"))*) --
                  -- Constraint: a given ATTNAME or #CONTENT can occur
                     only once; types apply to ultimate object of
                     address, not including any intermediate location
                     Address elements. --
                  -- Constraint: model groups limited to repeating
                     or non-repeating OR groups if refmodel option not
                     supported. Tokens in model groups must be GIs. --
      "#ALL #ANY" -- Constant --
>
]]><!-- reftype -->


<!entity % refctl "IGNORE">

               <!-- Reference Resolution Control -->
<![ %refctl; [
<!attlist
-- refctl --      -- Reference resolution control attributes --
                  -- Clause: 7.7.3 --
   #ALL

   refrange       -- Reference resolution range --
                  -- Clause: 7.7.2 --
                  -- B  Backward reference to identified local element
                     D  Identified local element, before or after
                     I  Location address allowed
                     X  Indirect target, may be external to reference
                     --
      CDATA       -- Lextype: (("#ALL",("B"|"D"|"I"|"X"))?,
                               (ATTORCON,("B"|"D"|"I"|"X"))*) --
                  -- Constraint: a given ATTNAME or #CONTENT
                     can occur only once. Its declared value
                     or lextype must contain IDREF. --
      "#ALL X"    -- Constant --

   reflevel       -- Reference resolution level --
                  -- Level that IDREF resolution cannot exceed,
                     relative to this element. --
      CDATA       -- Lextype: (("#ALL",unzi)?,(ATTORCON,unzi)*) --
                  -- Constraint: a given ATTNAME or #CONTENT
                     can occur only once. Its declared value
                     or lextype must contain IDREF. --
      #IMPLIED    -- Default: resolve fully --
>
]]><!-- refctl -->


<!entity % refloc "IGNORE">

                 <!-- Reference Location Address -->
<![ %refloc; [
<!entity % rflocatt '
-- refloc --      -- Reference Location Address --
                  -- Clause: 7.8 --
   #ALL

   loctype        -- Reference location addresses type --
                  -- Each named attribute treated as if it were an
```

```
                     IDREF to a location address element. --
                  -- Constraint: The declared values of named
                     attributes must be lexically compatible with
                     their specified interpretation. --
                  -- Note: The declared value CDATA always meets this
                     requirement. --
      CDATA       -- Lextype: (ATTORCON,("IDLOC"|"TREELOC"|
                                         "PATHLOC"|"RELLOC"|
                                         ("QUERYLOC",NOTATION)))+ --
      #IMPLIED    -- Constant --
                  -- Default: all references use SGML IDREFs, and each
                     IDREF in an IDREFS attribute is considered
                     separately --

   rflocsrc       -- Reference location source --
                  -- Associates referential attributes with their
                     location sources. --
      CDATA       -- Lextype: (ATTORCON,ATTORCON)+ --
                  -- Constraint: attributes named must be referential
                     attributes. --
      #IMPLIED    -- Constant --
                  -- Default: all referential attributes have this
                     element as their location source. --
'>
<!attlist %rflocatt; >
<!attlist #NOTATION %rflocatt; >
]]><!-- refloc -->

                   <!-- Reference Location Span -->
<![ %refloc; %spanloc; [
<!entity % rfspnatt '
-- rflocspn --    -- Reference location span --
                  -- Clause: 7.8 --
   #ALL

   rflocspn       -- Reference location span --
                  -- Names pairs of referential attributes that
                     address spans when both attributes are
                     specified. --
      CDATA       -- Lextype: (ATTORCON,ATTORCON)+ --
                  -- Constraint: attributes named must be referential
                     attributes. --
      #IMPLIED    -- Constant --
'>
<!attlist %rfspnatt; >
<!attlist #NOTATION %rfspnatt; >
>
]]><!-- refloc, spanloc -->


<!entity % proplat "IGNORE">

                <!-- Property Location Attributes -->
<![ %proplat; [
<!attlist
-- proplat --     -- Property location attributes --
```

```
                  -- Clause: 7.9.2 --
   (nmsploc,proploc)

   apropsrc       -- Use additional property source? --
      (apropsrc|solesrc)
      solesrc

   notprop        -- If not a property of locsrc or apropsrc? --
      NAME        -- Lextype: ("ERROR"|"IGNORE") --
      ERROR

   direct         -- Direct or indirect value --
                  -- Constraint: Ignored if no indirect value --
      (direct|indirect)
      indirect
>
]]><!-- proplat -->


<!entity % mixedloc "IGNORE">

                   <!-- Mixed Location Address -->
<![ %mixedloc; [
<!element
   mixedloc       -- Mixed location address --
                  -- Clause: 7.9.4 --
                  -- Groups location addresses together. --
   - O
   (%loc;)*

-- OptionalAttributes [locs]: multloc, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
]]><!-- mixedloc -->


<!entity % treecom "IGNORE">

                 <!-- Tree Combination Attributes -->
<![ %treecom; [
<!attlist
-- treecom --     -- Tree combination attributes --
                  -- Clause: 7.10.1.3 --
   (pathloc,treeloc)

   treecom        -- Tree combination --
                  -- Combine multiple source trees --
      (ntreecom|treecom)
      ntreecom
>
]]><!-- treecom -->

                  <!-- HyTime Lexical Tokenizer -->
```

```
<![ %HyLexTok; [
<!notation
   HyLexTok       -- HyTime Lexical Tokenizer --
                  -- A data tokenizer grove constructor using the
                     HyLex syntax. --
                  -- Clause: 7.10.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION HyTime Lexical Tokenizer//EN"

-- Attributes [locs]: HyLexTok --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyLexTok       -- HyTime Lexical Tokenizer --
                  -- Clause: 7.10.2 --

   HyBase    NAME     #FIXED datatok
   superdcn  NAME     #FIXED HyLex

-- Attributes from datatok notation --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED

-- Attributes from HyLex notation --
   norm     (norm|unorm) norm
>
<!attlist
-- dlhylex --     -- HyLex attributes for dataloc and datatok --
                  -- Clause: 7.10.2 --
   (dataloc,datatok)

-- Attributes from HyLex notation --
   norm     (norm|unorm) norm
>
<!entity % HyLex "INCLUDE">
]]><!-- HyLexTok -->

               <!-- Data Tokenizer Grove Definition -->
<![ %datatok; [
<!notation
   HyDatTok       -- HyTime Data Tokenizer --
                  -- A simple data tokenizer grove constructor. --
                  -- Clause: 7.10.2 --
                  -- Lextype: ("LINE"|"NAME"|"NORM"|"SINT"|"STR"|
                               "UTC"|"UTCDATE"|"UTCTIME"|"WORD") --

   PUBLIC "ISO/IEC 10744:1997//NOTATION HyTime Data Tokenizer//EN"

-- Attributes [locs]: HyDatTok --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   HyDatTok       -- HyTime Data Tokenizer --
                  -- Clause: 7.10.2 --

   HyBase    NAME     #FIXED datatok

-- Attributes from datatok form --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
<![ %HyLexTok; [
   <!entity % ddattok "HyLexTok">
]]>
<!entity %
   ddattok        -- Default data tokenizer notation --
                  -- Clause: 7.10.2 --

   "HyDatTok"
>
<!element
   datatok        -- Data tokenizer grove definition --
                  -- Clause: 7.10.2 --
                  -- Generates a datatok grove from data in another
                     grove. --
   - O
   (#PCDATA)

-- Attributes [locs]: datatok --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   datatok        -- Data location address --
                  -- Clause: 7.10.2 --

   HyBase   NAME     #FIXED agrovdef

-- Attributes from agrovdef --
   grovesrc CDATA    #REQUIRED
   grovecon NOTATION (HyDatTok|HyLexTok) #FIXED %ddattok;

-- Attributes from datatok notation form (via DAFE) --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
]]><!-- datatok -->
```

```
          <!-- Data Tokenizer Grove Construction Process -->
<![ %datatok; [
<!notation
   datatok        -- Data tokenizer grove construction process --
                  -- Clause: A.4.4.2.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Data Tokenizer Grove Construction Process//EN"

-- Attributes [locs]: datatok --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!attlist #NOTATION
   datatok        -- Data tokenizer grove construction process --
                  -- Clause: A.4.4.2.2 --

   catsrc         -- Source concatenation separator --
      CDATA
      #IMPLIED    -- Default: no source concatenation --

   cattoken       -- Token concatentation separator --
      CDATA
      #IMPLIED    -- Default: no token concatenation --

   catres         -- Result concatenation separator --
      CDATA
      #IMPLIED    -- Default: no result concatenation --

   boundary       -- Hit boundary constraint --
      (sodeod|sodiec|isceod|isciec|inmodel)
      isciec

   maxtoksz       -- Maximum token size --
      NUMBER
      #IMPLIED    -- Default: system defined --
>
]]><!-- datatok -->

                    <!-- Data Location Address -->
<![ %dataloc; [
<!element
   dataloc        -- Data location address --
                  -- Clause: 7.10.2 --
                  -- Locates string and token data objects in data --
   - O
   (%dimlist;)*   -- Constraint: interpreted as a single list of
                     dimension specifications --

-- Attributes [base]: overrun --
-- Attributes [locs]: dataloc, locsrc, impsrc --
-- OptionalAttributes [locs]: multloc, referatt, spanloc, treetype --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
```

```
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   dataloc        -- Data location address --
                  -- Clause: 7.10.2 --

   HyBase   NAME     #FIXED datatok
   HyBnames CDATA    #FIXED "HyBase #DEFAULT
                             #ARCCONT filter
                             grovesrc locsrc"

   filter         -- Tokenization filter --
      CDATA       -- Constraint: If no data tokenizer notation
                     specified, filter must be a single lexical type
                     name. --
      "str"

   tokordat        -- Result to return: tokens or source data? --
      (data|tokens)
      data

-- Attributes from datatok form --
   catsrc    CDATA    #IMPLIED
   cattoken  CDATA    #IMPLIED
   catres    CDATA    #IMPLIED
   boundary  (sodeod|sodiec|isceod|isciec|inmodel) isciec
   maxtoksz  NUMBER   #IMPLIED
>
<!entity % locsrc "INCLUDE">
<!entity % impsrc "INCLUDE">
<!entity % overrun "INCLUDE">
]]<!-- dataloc -->

<!entity % bibloc "IGNORE">

               <!-- Bibliographic Location Address -->
<![ %bibloc; [
<!element
   bibloc         -- Bibliographic location address --
                  -- Clause: 7.12 --
                  -- Bibliographic reference to real object.
                     Represents an address that is processable only as
                     data.  Always returns itself (the bibloc
                     element). --
   - O
   (%HyCFC;)*

-- Attributes [locs]: bibloc --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [locs]: bibloc:bibsrc --
>
<!attlist
```

```
   bibloc         -- Bibliographic location address --
                  -- Clause: 7.12 --

   bibsrc         -- Bibliographic source. Another bibloc element that
                     describes the context to which this bibloc is to
                     be applied. --
      CDATA       -- Reference --
                  -- Reftype: bibloc* --
      #IMPLIED
>
]]><!-- bibloc -->

                    <!-- Traversal Attributes -->
<![ %traverse; [
<!attlist
-- traverse --    -- Traversal attributes --
                  -- Clause: 8.1.3 --
   (agglink,hylink,ilink)

   linktrav       -- Hyperlink traversal rules --
                  -- Traversal between anchors of hyperlinks:
                       A  any traversal or departure (EID)
                       D  departure after internal arrival
                       E  traversal after external arrival
                       I  traversal after internal arrival
                       N  no traversal after internal arrival
                       P  no internal arrival
                       R  return traversal after internal arrival --
      NAMES       -- Lextype: ("A"|"EI"|"ER"|"ED"|"EN"|"EP"|"ERD"|
                               "I"|"ID"|"D"|"N"|"P"|"R"|"RD")+ --
                  -- Constraint: one per anchor or one for all --
      A

   listtrav       -- List traversal rules --
                  -- Traversal between members of list anchors:
                       A  adjacent (both left and right) traversal
                       L  left traversal
                       N  no traversal
                       R  right traversal
                       W  wrapping traversal --
      NAMES       -- Lextype: ("A"|"AW"|"L"|"LW"|"N"|"R"|"RW")+ --
                  -- Constraint: one per list anchor or one for all
                     list anchors --
      N
>
]]><!-- traverse -->


<!entity % hylink "IGNORE">

                   <!-- Hyperlink Relationship -->
<![ %hylink; [
<!element
   hylink         -- Hyperlink relationship --
                  -- Clause: 8.2.1 --
   - O
   (%HyCFC;)*

-- Attributes [links]: hylink --
-- OptionalAttributes [links]: traverse --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   hylink         -- Hyperlink relationship --
                  -- Clause: 8.2.1 --

   anchrole       -- Anchor roles  --
      CDATA       -- Lextype: (NAME,("#LIST"|"#CORLIST")?)+ --
                  -- Constraint: all #CORLIST anchors of a link must
                     have the same number of members. --
      #REQUIRED   -- Constant --

   anchcstr       -- Anchor existence constraints --
      NAMES       -- Lextype: ("REQUIRED"|"SELF"|"OMIT"|"COND")+ --
                  -- Constraint: one per anchor or one for all --
      REQUIRED    -- Constant --

   emptyanc       -- Is empty anchor an error? --
      NAMES       -- Lextype: ("ERROR"|"NOTERROR")+ --
                  -- Constraint: one per anchor or one for all --
      ERROR

-- Anchor addressing attributes: one per non-self anchor role, with
   same name as anchor role (unless remapped by the architectural
   attribute renamer attribute). --
>
]]><!-- hylink -->

                  <!-- Contextual Link Traversal -->
<![ %clink; %traverse; [
<!attlist
-- clinktra --    -- Contextual link traversal --
                  -- Clause: 8.2.2 --
   (clink)

   linktrav CDATA    "A ID"
>
]]><!-- clink, traverse -->


<!entity % varlink "IGNORE">

            <!-- Variable Link -->
<![ %varlink; [
<!element
   varlink        -- Variable link --
                  -- Clause: 8.2.4 --
   - O
   (anchspec)+
-- Attributes [links]: ancspcat, varlink, vartrav --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!element
   anchspec       -- Anchor specification --
                  -- Clause: 8.2.4 --
   - O
   (%HyCFC;)*     -- Reference --
                  -- Note: Use of refloc facility required --

-- Attributes [links]: anchspec, ancspcat, vartrav --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   anchspec       -- Anchor specification --
                  -- Clause: 8.2.4 --

   multmem        -- May anchor have multiple members? --
      (single|list|corlist)
      single

   emptyanc       -- Is empty anchor an error? --
      (error|noterror)
      error
>

<!attlist
-- ancspcat --    -- Anchor specification attributes --
                  -- Clause: 8.2.4 --
   (anchspec,varlink)

   anchrole       -- Anchor role --
      NAME
      #IMPLIED    -- Default: for anchspec, anchor role is GI of element --
                  -- Default: for varlink, varlink is not self anchor --
>
<![ %traverse; [
<!attlis
-- vartrav --     -- Varlink traversal rules --
                  -- Constraint: ignored on varlink element if it is
                     not a self anchor --
   (anchspec,varlink)

   linktrav       -- Hyperlink traversal rules --
                  -- Traversal between anchors of hyperlinks:
                       A  any traversal or departure (EID)
                       D  departure after internal arrival
                       E  traversal after external arrival
                       I  traversal after internal arrival
                       N  no traversal after internal arrival
                       P  no internal arrival
                       R  return traversal after internal arrival --
      (A|EI|ER|ED|EN|EP|ERD|I|ID|D|N|P|R|RD)
      A

   listtrav       -- List traversal rules --
                  -- Traversal between members of list anchors:
                       A  adjacent (both left and right) traversal
                       L  left traversal
                       N  no traversal
                       R  right traversal
                       W  wrapping traversal --
      (A|AW|L|LW|N|R|RW)
                  -- Constraint: ignored if anchor is not a list --
      N
>
]]><!-- traverse -->
]]><!-- varlink -->


<!entity % ilink "IGNORE">

                      <!-- Independent Link -->
<![ %ilink; [
<!element
   ilink          -- Independent link --
                  -- Clause: 8.2.5 --
   - O
   (%HyCFC;)*

-- Attributes [links]: ilink --
-- OptionalAttributes [links]: traverse --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   ilink          -- Independent link --
                  -- Clause: 8.2.5 --

   anchrole       -- Anchor roles --
      CDATA       -- Lextype: (NAME,("#AGG"|"#LIST"|"#CORLIST")?)+ --
                  -- Constraint: one per anchor --
      #REQUIRED   -- Constant --

   linkends       -- Link ends --
      IDREFS      -- Reference --
                  -- Constraint: One ID per anchor role, except
                     that the first ID may be omitted if the first
                     anchor is the link element itself. --
      #REQUIRED
>
]]><!-- ilink -->
```

```
<!entity % HyGrand "IGNORE">

             <!-- HyTime Granule Definition Notation -->
<![ %HyGrand; [
<!notation
   HyGrand        -- HyTime Granule Definition Notation --
                  -- Clause: 9.2.2 --
                  -- The mathematical relationship between a granule
                     and its measurement domain's SMU is specified as
                     a ratio of the granule being defined to either
                     the SMU or another granule of the same domain. --
                  -- Lextype: (ratio,(granule|SMU)) --
                  -- Constraint: A granule may not be defined in terms of
                     itself, either directly or indirectly. --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Granule Definition Notation//EN"

-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<!entity % measure "INCLUDE">
]]><!-- HyGrand -->


<!entity % measure "IGNORE">

                <!-- Measurement Domain Definition -->
<![ %measure; [
<![ %HyGrand; [
   <!entity % dgdnot "HyGrand">
]]>
<!entity %
   dgdnot         -- Default granule definition notation --
                  -- Clause: 9.2.1 --

   "#REQUIRED"
>
<!element
   measure        -- Measurement domain definition --
                  -- Clause: 9.2.1 --
                  -- Defines a set of granules in terms of a standard
                     measurement unit (SMU).  Multiple measurement
                     domain definitions for the same SMU comprise one
                     domain. --
   - O
   (granule+)

-- Attributes [sched]: measure --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
```

```
<!attlist
   measure        -- Measurement domain definition --
                  -- Clause: 9.2.1 --

   smu            -- Standard measurement unit --
                  -- The standard measurement unit for the domain --
      NAME        -- Lextype: SMU --
      #REQUIRED
>
<!element
   granule        -- Granule definition --
                  -- Clause: 9.2.1 --
                  -- The content of a granule definition element
                     defines the mathematical relationship between the
                     granule being defined and the measurement domain's
                     SMU. --
                  -- Note: The relationship may be defined in terms of
                     another granule of the same measurement domain. --
   - O
   (%HyCFC;)*

-- Attributes [sched]: granule --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>
<!attlist
   granule        -- Granule definition --
                  -- Clause: 9.2.1 --

   gn             -- Granule name --
                  -- The granule name being defined. Granule names are
                     not normalized as SGML names (that is, they are
                     case-sensitive). --
      CDATA       -- Lextype: granule --
                  -- Constraint: unique within measurement domain --
                  -- Constraint: cannot be same as SMU name (with case
                     ignored) --
      #REQUIRED

   gdnot          -- Granule definition notation --
                  -- Notation used to specify the relationship between
                     the granule being defined and the measurement
                     domain's SMU. --
      NAME        -- Lextype: NOTATION --
      %dgdnot;
>
]]><!-- measure -->

                      <!-- Measurement Units -->
<![ %sched; [
<!notation
   gQuantum       -- Generic quantum --
                  -- Reference unit of generic quanta --
   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Virtual Measurement Unit//EN"
>
<!notation
   SIsecond       -- Systeme International second --
                  -- Reference unit of real time --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International second//EN"
>
<!notation
   SImeter        -- Systeme International meter --
                  -- Reference unit of real length --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International meter//EN"
>
<!notation
   virTime        -- Virtual Time --
                  -- Reference unit of virtual time --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Virtual Measurement Unit//EN"
>
<!notation
   virSpace       -- Virtual Space --
                  -- Reference unit of virtual space --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Virtual Measurement Unit//EN"
>
]]><!-- sched -->

<![ %sched; [
<!notation
   SIkg           -- Systeme International kilogram --
                  -- Reference unit of mass --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International kilogram//EN"
>
<!notation
   SIkelvin       -- Systeme International kelvin --
                  -- Reference unit of thermodynamic temperature --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International kelvin//EN"
>
<!notation
   SIcd           -- Systeme International candela --
                  -- Reference unit of luminous intensity --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International candela//EN"
>
<!notation
```

```
   SIampere       -- Systeme International ampere --
                  -- Reference unit of electric current --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International ampere//EN"
>
<!notation
   SImole         -- Systeme International mole --
                  -- Reference unit of amount of substance --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International mole//EN"
>
<!notation
   SIradian       -- Systeme International radian --
                  -- Reference unit of plane angle --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International radian//EN"
>
<!notation
   SIsr           -- Systeme International steradian --
                  -- Reference unit of solid angle --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International steradian//EN"
>
]]><!-- sched -->

<![ %sched; [
<!notation
   SIC            -- Systeme International degree Celsius --
                  -- Reference unit of celsius temperature --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Systeme International degree Celsius//EN"
>
]]><!-- sched -->

<![ %sched; [
<!notation
   CHNYUAN        -- Yuan currency unit of China --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Yuan currency unit of China//EN"
>
<!notation
   GBRPOUND       -- Pound currency unit of United Kingdom --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           Pound currency unit of United Kingdom//EN"
>
]]><!-- sched -->

                   <!-- Finite Coordinate Space -->
<![ %sched; [
<!element
   fcs            -- Finite coordinate space --
                  -- Clause: 9.3 --
   - O
   (baton|evsched|wand)*

-- Attributes [sched]: fcs --
-- OptionalAttributes [sched]: calibrat --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, pdimref:prjtarg,
   rfcsloc:prjsrc, sched:fcs --
>
<!attlist
   fcs            -- Finite coordinate space --
                  -- Clause: 9.3 --

   axes           -- Axis names and measurement domains --
                  -- The name and measurement domain (SMU) of each
                     axis. --
      CDATA       -- Lextype: (AXISNM,SMU)+ --
                  -- Constraint: Each axis name must be unique
                     within the list. --
      #REQUIRED   -- Constant --

-- Each axis name specified in the value of the axes attribute is
   taken to be the name of an attribute of the client element that
   specifies the dimension and granularity of the axis.  Each such
   attribute has the lexical type (NUMBER, granule?, ("#MDU", ratio,
   granule?)?), where the number is the dimension of the axis,
   expressed in terms of the following granule, if specified;
   otherwise the dimension is in terms of the axis SMU.  If a granule
   is specified, it is both the default base granule and default HMU
   for the axis; otherwise the default base granule and HMU is the
   axis SMU.  If #MDU and a ratio is specified, it defines the MDU to
   SMU ratio for the axis, specified in terms of either the following
   granule or the default HMU.  All granule names must be defined in
   terms of the axis SMU. --
>
]]><!-- sched -->

                          <!-- Schedule -->
<![ %sched; [
<!attlist
-- sched --       -- Schedule --
                  -- Clause: 9.4.1 --
   (baton,evsched,wand)

   fcs            -- Governing finite coordinate space element --
      CDATA       -- Reference --
                  -- Reftype: fcs --
      #IMPLIED    -- Default: containing fcs element --
                  -- Constraint: required if schedule is not contained
                     by fcs element. --
   axisord        -- Axis order --
                  -- The order of axes for extent specification within
                     schedule.  For each axis name specified, an
                     attribute of the same name may be used to specify
                     the HMU for the axis.  The lexical type of such
                     attributes is (ratio, granule?), where the ratio
                     is the number of HMUs per base granule, and the
                     granule, if specified, is the base granule.  If a
                     granule is not specified, the default base
                     granule defined by the governing fcs element is
                     used.  A specified granule must be defined in
                     terms of the axis SMU.  If a value is not
                     supplied for an axis HMU specification attribute,
                     the HMU for that axis is the default HMU defined
                     by the governing fcs element. --
      CDATA       -- Lextype: (AXISNM)+ --
                  -- Constraint: axis names must be defined by
                     governing fcs element --
      #IMPLIED    -- Constant --
                  -- Default: Axis order is unchanged from that
                     specified by governing fcs element, and each axis
                     may have an HMU specification attribute. --

   sorted         -- Extent ordering --
                  -- Ordering of elements with respect to the ordering
                     of their scheduled extents. --
      (sorted|unsorted)
      unsorted

   coverage       -- Extent coverage --
                  -- Whether or not gaps may occur between the
                     scheduled extents of elements in the schedule. --
      (solid|sparse)
      sparse

   overlap        -- Extent overlap --
                  -- Whether or not scheduled extents of elements in
                     the schedule may overlap. --
                  -- Constraint: If the schedule is a wand, the value
                     is ignored and is always treated as noverlap. --
      (overlap|noverlap)
      #IMPLIED    -- Default: If the schedule is not a wand, the
                     default value is overlap. --
>
]]><!-- sched -->

                    <!-- Extent Specification -->
<![ %sched; [
<!attlist
-- exspec --      -- Nominal extent specification --
                  -- Clause: 9.4.2 --
   (event,modscope,proscope)

   exspec         -- Extent specification --
      CDATA       -- Reference --
```

```
                  -- Reftype: (extent|extlist)+ --
      #REQUIRED
>
]]><!-- sched -->

                           <!-- Extent -->
<![ %sched; [
<![ %HyExSpec; [
   <!entity % dexspec "HyExSpec">
]]>
<!entity %
   dexspec        -- Default extent specification notation --
                  -- Clause: 9.4.5 --
   "#IMPLIED"
>
<!element
   extent         -- Extent --
                  -- Clause: 9.4.4 --
                  -- A position and quantum count along one or more
                     axes. --
   O O
   (%HyCFC;|%dimlist;)*
                  -- Constraint: if no extent specification notation
                     specified, then content is restricted to
                     (dimspec)* --

-- Attributes [sched]: extent --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, exspec:exspec, grpdex:grpdex,
   grprepet:repscope --
>
<!attlist
   extent         -- Extent --
                  -- Clause: 9.4.4 --

   notation       -- Extent specification notation --
      NAME        -- Lextype: NOTATION --
      %dexspec;   -- Default: content is restricted to (dimspec)* --
>
]]><!-- sched -->

                         <!-- Extent List -->
<![ %sched; [
<![ %HyExtLst; [
   <!entity % dextlist "HyExtLst">
]]>
<!entity %
   dextlist       -- Default extent list notation --
                  -- Clause: 9.4.5 --

   "#IMPLIED"
>
<!element
```

```
   extlist        -- Extent List --
                  -- Clause: 9.4.5 --
                  -- A list of extents on one or more axes. --
   O O
   (%HyCFC;|%dimlist;|extent|extlist)*
                  -- Constraint: if no extent list notation
                     specified, then content is restricted to
                     (extent|extlist)* --

-- Attributes [sched]: extlist
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, exspec:exspec, grpdex:grpdex --
>
<!attlist
   extlist        -- Extent List --
                  -- Clause: 9.4.5 --

   notation       -- Extent list notation --
      NAME        -- Lextype: NOTATION --
      %dextlist;  -- Default: content is restricted to
                     (extent|extlist)* --
>
]]><!-- sched -->

                       <!-- Event Schedule -->
<![ %sched; [
<!element
   evsched        -- Event schedule --
                  -- Clause: 9.5 --
   - O
   (event|evgrp)*

-- Attributes [sched]: sched --
-- OptionalAttributes [sched]: pulsemap --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id,
   ireftype, lextype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [sched]: dimref:elemspec, dimref:schdspec,
   pdimref:prjtarg, pulsemap:pulsemap, rfcsloc:prjsrc --
-- Referrers [rend]: batrule:scheds, batrule:targschd, modrule:events,
   prorule:sources, prorule:targschd, wandrule:evscheds --
>
]]><!-- sched -->

                <!-- Projected Dimension Reference -->
<![ %dimref; %project; [
<!attlist
-- pdimref --     -- Projected dimension reference --
                  -- Clause: 9.8.2.2 --
                  -- Dimref attributes for references to projected
                     sets of events, modscopes and/or proscopes --
   (dimref)
   prjdirct       -- Projected and/or direct --
                  -- Selection on the basis of whether events,
                     modscopes, or proscopes were either projected
                     onto, or directly scheduled in, the specified
                     element.
                        drctonly: Direct only
                        projonly: Projected only
                        drctproj: Direct and projected --
                  -- Constraint: Value is ignored if element specified
                     is not evsched, wand, baton, or fcs. --
                  -- Constraint: Value is ignored (is, in effect,
                     drctonly) if a value is specified for extnum. --
      (drctonly|drctproj|projonly)
      drctonly

   prjtarg        -- Projected to target --
                  -- Referenced dimension exists by projection onto
                     element specified by prjtarg attribute. --
                  -- Constraint: Value is ignored (and referenced
                     dimension is not a projected dimension) if
                     element specified is extent or extlist. --
      CDATA       -- Reference --
                  -- Reftype: (baton|evsched|fcs|wand) --
      #IMPLIED    -- Default: Referenced dimension is unprojected. --

   prjby          -- Projected by --
                  -- Referenced dimension exists by projection onto
                     element specified by prjtarg attribute only by
                     elements specified by prjby attribute. --
                  -- Constraint: Value is ignored unless prjtarg has
                     non-ignored value. --
      CDATA       -- Reference --
                  -- Reftype: (baton|progrp|prorule|proscope) --
      #IMPLIED    -- Default: All relevant prorules, batons, progrps,
                     and proscopes will be taken into account. --
>
]]><!-- dimref, project -->

<![ %fcsloc; %project; [
<!attlist
-- rfcsloc --     -- Finite coordinate space location address with
                     rendition module --
                  -- Clause: 7.10.2 --
   (fcsloc)

   prjdirct       -- Projected and/or direct --
                  -- Selection based on how objects were scheduled:
                      drctonly: (Direct only) Do not select the
                                events, modscopes, and proscopes
                                projected onto the element specified.
                      projonly: (Projected only) Do not select the
                                events, modscopes, and proscopes
                                directly scheduled in or by the
                                element specified.
                      drctproj: (Direct and projected) Do not exclude
```

```
                                direct or projected events, modscopes,
                                and proscopes from being selected. --
      (drctonly|drctproj|projonly)
      drctonly

   prjsrc         -- Projection sources --
                  -- Of the events, modscopes and proscopes indirectly
                     scheduled on the locsrc elements, select only
                     those that are directly scheduled by or in the
                     specified elements. --
      CDATA       -- Reference --
                  -- Reftype: (baton|event|evgrp|evsched|fcs|modgrp|
                               modscope|progrp|proscope|wand)* --
                  -- Constraint: Value is ignored if value of prjdrct
                     is drctonly. --
      #IMPLIED    -- Default: No projection sources constraint. --

   prjby          -- Projected by --
                  -- Of the events, modscopes and proscopes indirectly
                     scheduled on the locsrc elements, select only
                     those that are projected there by the specified
                     projection elements. --
      CDATA       -- Reference --
                  -- Reftype: (baton|progrp|prorule|proscope)* --
                  -- Constraint: Value is ignored if value of prjdrct
                     is drctonly. --
      #IMPLIED    -- Default: No projection elements constraint. --
>
]]><!-- fcsloc, project -->

            <!-- Project modified or unmodified object -->
<![ %modify; %project; [
<!attlist
-- modified --    -- Project modified or unmodified object --
                  -- Clause: 10.3.2.1 --
   (batrule|prorule)

   modified       -- Project event (or modscope) with object (or
                     modifier) modified or unmodified --
                  -- Project modified objects (or modifiers)
                     ("modify"), or project objects (or modifiers)
                     without the modifications that are applied to
                     them ("unmodify") in their source schedules. --
                  -- Constraint: value is ignored for projected
                     elements that are not events or modscopes. --
      (modify|unmodify)
      modify
>
]]><!-- modify, project -->
```

C.3 General Architecture Meta-Declarations

The General Architecture meta-declaration set is:

```
<!AFDR "ISO/IEC 10744:1997">
```

```
<!--
"ISO/IEC 10744:1997//DTD AFDR Meta-DTD
 General Architecture//EN"
-->

<!--

Option Summary:

altreps   Alternative Representation Facility
dafe      Data Attributes for Elements Facility
dvlist    Default Value List Facility
HyLex     HyTime Lexical Model Notation
HyOrd     HyTime Lexicographic Ordering Notation
included  Included Entities Facility Facility
ireftype  Immediate ID Reference Type Facility
lextype   Lexical Typing Facility
opacity   Element Opacity Facility
REGEX     POSIX Regular Expression Notation
superdcn  Notation Derivation Source Facility

-->

<?IS10744 USELEX SGMLlex>
<!entity %
   SGMLlex        -- SGML lexical types --

   PUBLIC "ISO/IEC 10744:1997//NONSGML LTDR LEXTYPES
           SGML Lexical Types//EN"
>

    <!-- General Architecture Content Model Parameter Entities -->
<!entity %
   GACFC          -- General Architecture context-free content --
   "GABrid"
>
<!entity %
   GAResrc        -- General Architecture resource forms --
   "dvlist"
>

            <!-- General Architecture Document Element -->
<!element
   GADoc          -- General Architecture document element --
   - O
   (%GACFC;)*
   +(%GAResrc;)
>

            <!-- General Architecture Bridging Element -->
<!element
   GABrid         -- General Architecture bridging element --
   - O
   (%GACFC;)*
>
```

```
           <!-- General Architecture Bridging Notation -->
<!notation
   GABridN        -- General Architecture bridging notation --
   PUBLIC
      "ISO/IEC 10744:1997//NOTATION AFDR ARCDATA
       General Architecture Bridging Notation//EN"
>
                      <!-- Unique Identifier -->
<!attlist
-- id --          -- Unique identifier --
                  -- Clause: A.5.2 --
   #ALL

   id             -- Unique identifier --
      ID
      #IMPLIED    -- Default: none --
>

                   <!-- Element Type Full Name -->
<!attlist
-- etfullnm --    -- Element type full name --
                  -- Clause: A.5.2 --
   #ALL

   etfullnm       -- Element type full name --
     CDATA
     #IMPLIED     -- Default: generic identifier --
>


<!entity % opacity "IGNORE">

                           <!-- Opacity -->
<![ %opacity; [
<!attlist
-- opacity --     -- Opacity --
                  -- Clause: A.5.2 --
   #ALL

   opacity        -- Opacity --
                  -- Transparent or opaque element --
      (transpar|opaque)
      opaque      -- Constant --
>
]]><!-- opacity -->


<!entity % dafe "IGNORE">

                <!-- Data attributes for elements -->
<![ %dafe; [
<!Attlist
-- dafe --        -- Data attributes for elements --
                  -- Clause: A.5.3.1 --
   #ALL
   NotNames       -- Data attributes for elements renamer --
                  -- Defines user names for data attributes --
      CDATA       -- Lextype: ((NAME,(ATTORCON|"#DEFAULT"),
                                 ("#MAPTOKEN",NMTOKEN,NMTOKEN)*)|
                                ("#NOTCONT",ATTNAME))* --
                  -- Constraint: a given ATTNAME, NAME, #CONTENT, or
             #NOTCONT can occur only once --
                  -- Constraint: data attribute name precedes user
                     name --
      #IMPLIED    -- Constant --
                -- Default: no renaming --

   NotSupr        -- Data attributes for elements suppressor --
                  -- Suppress data attribute for elements
                     processing --
      (sNotAll|sNotForm|sNotNone)
      #IMPLIED    -- Default: inherited --
>
]]><!-- dafe -->


<!entity % lextype "IGNORE">

                  <!-- Lexical Typing Attribute -->
<![ %lextype; [
<!attlist
-- lextype --     -- Lexical typing attribute --
                  -- Clause: A.5.4 --
   #ALL

   lextype        -- Lexical types --
                  -- Lexical types of attribute values or character
                     data content --
      CDATA       -- Lextype: (ATTORCON,NAME)* --
                  -- Constraint: a given ATTNAME or #CONTENT can occur
                     only once --
      #IMPLIED    -- Constant --
                  -- Default: none --
>
]]><!-- lextype -->


<!entity % ireftype "IGNORE">

             <!-- ID Immediate Referent Element Type -->
<![ %ireftype; [
<!attlist
-- ireftype --    -- ID immediate referent element type --
                  -- Clause: A.5.5 --
   #ALL

   ireftype       -- ID immediate referent element type --
      CDATA       -- Lextype: (("#ALL",(GI|modelgroup|"#ANY"))?,
                                (ATTORCON,(GI|modelgroup|"#ANY"))*) --
                  -- Constraint: a given ATTNAME or #CONTENT can occur
                     only once; types apply to immediate object of
                     address. --
                  -- Constraint: model groups limited to repeating
                     or non-repeating OR groups if irefmodel option not
                     supported. Tokens in model groups must be GIs. --
      "#ALL #ANY" -- Constant --
>
]]><!-- ireftype -->


<!entity % dvlist "IGNORE">

                <!-- Default Value List Attributes -->
<![ %dvlist; [
<!attlist
-- dvlatt --      -- Default value list attributes --
                  -- Clause: A.5.6.1 --
   #ALL

   subdvl         -- Subelement impliable attribute value defaults --
      IDREFS      -- Reference --
                  -- Reftype: dvlist+ --
                  -- Note: Cannot be indirect --
      #IMPLIED    -- Default: none --

   sibdvl         -- Sibling impliable attribute value defaults --
      IDREFS      -- Reference --
                  -- Reftype: dvlist+ --
                  -- Note: Cannot be indirect --
      #IMPLIED    -- Default: none --

   selfdvl        -- Self impliable attribute value defaults --
      IDREFS      -- Reference --
                  -- Reftype: dvlist+ --
                  -- Note: Cannot be indirect --
      #IMPLIED    -- Default: none --
>
]]><!-- dvlist -->

                     <!-- Default Value List -->
<![ %dvlist; [
<!element
   dvlist         -- Default value list --
                  -- Clause: A.5.6.2 --
   - O
   (#PCDATA)      -- Ulextype: attspecs --

-- Attributes: dvlist --
-- CommonAttributes: dafe, dvlist, etfullnm, id, ireftype,
   lextype, opacity --
-- Referrers: dvlatt:selfdvl, dvlatt:sibdvl, dvlatt:subdvl --
>
<!attlist
   dvlist         -- Default value list --
                  -- Clause: A.5.6.2 --

   id             -- Unique identifier --
```

```
      ID
      #REQUIRED

   dvgi           -- Default value element types --
                  -- Applies to all elements if omitted --
      CDATA       -- Lextype: (GI+|(#ALL,GI*)) --
      #IMPLIED    -- Default: all elements --

   preatts        -- Attributes whose values are to be preempted --
      NAMES       -- Constraint: must be in dvlist content --
      #IMPLIED    -- Default: none --

   defatts        -- Attributes whose values become the default value
                     when specified --
      NAMES       -- Constraint: must be in dvlist content --
      #IMPLIED    -- Default: no replaceable defaults --
>
]]><!-- dvlist -->


<!entity % included "IGNORE">

            <!-- Entities included from notation data -->
<![ %included; [
<!attlist #NOTATION
-- included --    -- Entities included from notation data --
                  -- Clause: A.5.7.1 --
   #ALL

   included       -- Entities included from notation data --
      CDATA       -- Lextype: ENTITIES --
      #IMPLIED    -- Default: no included entities --
>
]]><!-- included -->


<!entity % altreps "IGNORE">

                  <!-- Alternate Representations -->
<![ %altreps; [
<!attlist #NOTATION
-- altreps --     -- Alternate representations --
                  -- Clause: A.5.7.1 --
   #ALL

   altreps        -- Alternate representations --
                  -- Alternative representations of this entity --
      CDATA       -- Lextype: ENTITIES --
      #IMPLIED    -- Default: none --
>
]]><!-- altreps -->


<!entity % superdcn "IGNORE">

                 <!-- Notation Derivation Source -->
```

```
<![ %superdcn; [
<!attlist #NOTATION
-- superdcn --    -- Notation derivation source --
                  -- Clause: A.5.7.1 --
   #ALL

   superdcn       -- Notation derivation source --
                  -- Notation on which this one is based --
      NAME        -- Lextype: NOTATION --
      #IMPLIED    -- Default: none --
>
]]><!-- superdcn -->


<!entity % HyLex "IGNORE">

                <!-- HyTime Lexical Model Notation -->
<![ %HyLex; [
<!notation
   HyLex          -- HyTime lexical model notation --
                  -- Clause: A.2.2 --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime lexical model notation (HyLex)//EN"

-- Attributes: HyLex --
-- CommonAttributes: altreps, included, superdcn --
>
<!attlist #NOTATION
   HyLex          -- HyTime lexical model notation --
                  -- Clause: A.2.3 --

   norm           -- Normalization --
      (norm|unorm)
      norm
>
]]><!-- HyLex -->


<!entity % HyOrd "IGNORE">

      <!-- HyTime Lexicographic Ordering Definition Notation -->
<![ %HyOrd; [
<!notation
   HyOrd          -- HyTime lexicographic ordering definition
                     notation --

   PUBLIC "ISO/IEC 10744:1997//NOTATION
           HyTime Lexicographic Ordering Definition Notation
           (HyOrd)//EN"

-- CommonAttributes: altreps, included, superdcn --
>
]]><!-- HyOrd -->
```

```
<!entity % REGEX "IGNORE">

              <!-- POSIX Regular Expression Notation -->
<![ %REGEX; [
<!notation
   REGEX          -- POSIX regular expression notation --

   PUBLIC "ISO/IEC 9945-2:1997//NOTATION
           POSIX Regular Expression Notation//EN"
>
<!attlist #NOTATION
   REGEX          -- POSIX regular expression notation --

   case           -- Case sensitivity --
                  -- Case sensitive match (case) or case insensitive
                     match (icase) --
      (case|icase)
      case
>
]]><!-- REGEX -->
```

Annex D
(informative)

Supplementary materials

This annex lists tutorial and reference materials for use with HyTime.

The following publications are available in the open literature:

— Charles F. Goldfarb, “Hytime: A standard for structured hypermedia interchange”, *IEEE Computer*, vol.24, no.8, pp.81-84, August, 1991. A Japanese translation is published in *Nikkei Electronics* 1991.12.9 (no.542).

An introduction to the concept of integrated open hypermedia and HyTime's approach to it.

— Steven R. Newcomb, Neill A. Kipp, Victoria T. Newcomb, “The HyTime Hypermedia/Time-based Document Structuring Language”, *Communications of the ACM*, November, 1991.

An extensive illustrated tutorial on HyTime.

The following materials are distributed bv the SGML Users' Group, P.O. Box 361, Swindon, Wiltshire, SN5 7BF, United Kingdom, http://www.hytime.org/:[1]

— HyTime Executive Overview

A brief summary of what HyTime is and does.

— Catalog of HyTime Architectural Forms

A summary of all architectural forms, including their interrelationship, mandatory or non-mandatory status, and a cross-reference to their specification in this International Standard.

— HyTime SGML Specification

All of the formal SGML specifications that define HyTime, in machine-readable form.

— The HyTime Processing Model

A description of how a possible HyTime implementation would process each architectural form.

— HyTime Examples

Illustrative papers covering the location address and hyperlinking facilities, the scheduling facility, synchronization and alignment, and the use of HyTime for representing graphical user interface objects.

— How to Read an International Standard

A brief informal explanation of editorial conventions that are used in ISO standards.

— What to Tell Your Child About SGML

An informal introduction to SGML concepts.

[1] The SGML Users' Group is an international non-profit membership organization, chartered as an educational charity in the United Kingdom, and is a liaison member of the ISO/IEC subcommittee that developed HyTime. Nevertheless, the documents that it distributes have not been subject to ISO/IEC review procedures, have no official status, and are not endorsed by the ISO or IEC or any of its national member bodies or affiliates.

ICS

Descriptors: .

Price based on 450 pages.Prix base sur 450 pages.